Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú

UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL: Aplicación de Gestión de Gastos para Smartphones con Sistema Operativo Android

AUTOR: Alberto Pérez Méndez

TITULACIÓ: Enginyeria Tècnica en Informàtica de Gestió

DIRECTOR: Lluís Padró Cirera

DEPARTAMENT: LSI - Departament de Llenguatges i Sistemes Informàtics

DATA:

TÍTOL: Aplicación de Gestión de Gastos para Smartphones con Sistema Operativo Android

COGNOMS: Pérez Méndez **NOM:** Alberto

TITULACIÓ: Enginyeria Tècnica en Informàtica

ESPECIALITAT: Gestió **PLA:** 92

DIRECTOR: Lluís Padró Cirera

DEPARTAMENT: Llenguatges i Sistemes Informàtics

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA:

Aquest Projecte té en compte aspectes mediambientals: □ Sí □ No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

Este proyecto de final de carrera consiste en desarrollar una aplicación para el sistema operativo Android para gestionar los gastos personales del usuario, y poder mantener un control de los mismos.

Los gastos se podrán asignar por categorías y lugares para poder hacer dicha gestión, y a la vez realizar un control más eficaz de los gastos. En la aplicación se podrán gestionar dichas categorías y lugares.

La aplicación además permitirá mostrar información estadística y grafica de los gastos realizados, mediante distintas pantallas consiguiendo que el usuario pueda tener una mayor información sobre sus hábitos de gastos de una manera muy visual.

Para facilitar el uso de la aplicación al usuario se han añadido diversas funcionalidades, como por ejemplo, los lugares de gasto se podrán obtener de forma automática por geo localización o mediante la selección de lugar en un mapa, y así mismo se podrá añadir los datos de un gasto mediante reconocimiento óptico de caracteres, facilitando así su utilización para cualquier tipo de usuario.

Se ha realizado un estudio de otras aplicaciones de gestión de gastos para conocer el mercado y así poder valorar sus puntos fuertes y débiles, consiguiendo una aplicación con unas características que no tienen las del mercado actual.

Respecto a la plataforma de desarrollo se han analizado las distintas alternativas, y se ha estudiado la plataforma utilizada finalmente que ha sido Android, puesto que es una plataforma en crecimiento y de código libre, lo que la hace más interesante para el desarrollo de este proyecto.

Paraules clau (màxim 10):

Android	Gestión	Gastos	SQL
XML	Java	OCR	Google
Graficas			

ÍNDICE

1. INTRODUCCIÓN

1.1 JUSTIFICACIÓN DEL PROYECTO

La elección del tipo de proyecto final de carrera, y su temática fue elegida después de varias reuniones con el tutor del proyecto, Lluís Padró, así como la plataforma tecnológica en la que queríamos desarrollar el proyecto.

La temática, después de evaluar diferentes posibilidades decidimos que sería una **aplicación de Gestión de Gastos personales**, donde el usuario podría fácilmente desde su móvil seguir los gastos del día a día, y obtener estadísticas de los mismos.

La aplicación gestionará los gastos de un único usuario, que será el usuario actual del móvil; no queremos complicar la aplicación añadiendo múltiples cuentas, debido a que el usuario debe poder añadir un gasto de forma rápida, y simple sin preocuparse de otras cosas.

¿Por qué una aplicación de Gestión de Gastos? Porque la temática la quería orientar a una utilidad práctica, quería realizar un proyecto de algo que pudiera resultar útil y atractivo para cualquier persona que utilice un Teléfono Móvil con el Sistema Operativo Android.

La tecnología que hemos decidido utilizar fue elegida rápidamente, ya que ambos estábamos interesados en la plataforma Android, a mí, porque me parece una plataforma muy llamativa por su expansión en los últimos años, y a mi tutor, porque nunca había llevado el proyecto de una aplicación Android.

Como último punto a nombrar, decir también que en el apartado de tecnología, decidimos incluir una tecnología novedosa como es el **Reconocimiento Óptico de Caracteres** como una forma opcional para introducir datos en la aplicación. (*Ver Anexo 2, para información sobre la tecnología de Reconocimiento Óptico de Caracteres*).

1.2 ANÁLISIS DE OTRAS APLICACIONES DE GESTIÓN DE GASTOS

En la *Play Store de Android* hay diversas aplicaciones destinadas a la gestión de gastos, a continuación se detalla un pequeño análisis de las funcionalidades de algunas de estas aplicaciones, para ver los puntos positivos y negativos de cada una de ellas, las cuales sirvieron para tomar ideas de cómo realizar mi aplicación.

- ***EXPENSE MANAGER:***

Es una aplicación con muchas características y funcionalidades, con gestión de múltiples cuentas de usuario. El interfaz es antiguo y no se rige a los estándares de *Google*. Hay reportes entre los usuarios que tienen problemas de fiabilidad, ya que varios de los usuarios indican que han perdido su información. También decir que no es una aplicación fácil de usar debido a la cantidad de funcionalidades, y a su interfaz antiguo.

- ***EXPENSE MANAGER (MARKUS HINTERSTEINER):***

Esta aplicación tiene un interfaz muy moderno, siguiendo los estándares de *Google*. Pero tiene muy pocas funcionalidades, la parte de estadísticas de la aplicación es de pago. De esta aplicación tome la idea de asignar un color a cada categoría.

- ***CONTROL DE GASTOS:***

Respecto a las otras 2 aplicaciones examinadas esta tiene una buena cantidad de funcionalidades, y un interfaz bastante agradable. Está muy orientada a controlar el gasto relacionándolo con la nómina del usuario.

Como resumen, de estas aplicaciones se puede destacar que demasiadas funcionalidades puedes ser contraproducentes, ya que pueden perder al usuario, y dificultar el uso de la aplicación.

Respecto al interfaz de usuario, se ven grandes diferencias entre un interfaz moderno, que son más fácil de usar y que además muestra al usuario realmente la información que necesite, y uno basado en estándares antiguos que dificulta el uso de la aplicación.

1.3 OBJETIVOS

Poder aplicar los conocimientos obtenidos durante los estudios para realizar este proyecto, es una motivación muy importante, y el principal objetivo de este proyecto. Ver cómo se puede aplicar de forma práctica en un proyecto final de carrera los años de estudio, es una gran satisfacción para cualquier alumno.

Este proyecto nace con el objetivo de familiarizarme con el Sistema Operativo Android. La plataforma Android se ha extendido rápida y masivamente los últimos años entre los usuarios, obteniendo la mayoría del mercado actual, haciéndola muy atractiva respecto a otras plataformas como *IOS de Apple*, *Windows Phone de Microsoft, Blackberry de RIM o Symbian de Nokia*.

La decisión de utilizar la *Plataforma Android* para realizar el PFC, es para poder desarrollar, y utilizar diferentes aspectos de la misma, como son:

- Desarrollar el interfaz de usuario de una aplicación Android.
- Usar una Base de Datos desde una aplicación Android
- Utilizar diferentes recursos del Smartphone Android mediante las APIs del sistema, como es la Cámara de Fotos, o el servicio GPS.
- Utilizar las distintas formas de realizar procesos dentro de Android, como son las Actividades, y los Fragmentos.
- Usar o llamar otras aplicaciones dentro de la Actividad principal de la aplicación.

Los objetivos funcionales de la aplicación son los siguientes:

- Gestión de Gastos del Usuario
- Gestión de las Categorías de los Gastos.
- Gestión de lugares de los Gastos.
- Consultar los Gastos del Usuario.
- Realizar estadísticas de los Gastos.
- Exportar los gastos en formato CSV.

- Gestionar las Preferencias de la aplicación.
- Estudiar la entrada de información utilizando Reconocimiento Óptico de Caracteres

El principal objetivo de este proyecto es crear una aplicación con la que el usuario pueda, de forma fácil y simple, gestionar sus gastos del día a día. Y de esta forma facilitarle al usuario la gestión de los mismos.

La aplicación con un interfaz fácil de usar debe permitir al usuario poder gestionar sus gastos personales. El usuario podrá añadir los gastos, y modificarlos o eliminarlos después. Estos gastos se podrán añadir de forma manual, y también se le ofrecerá al usuario la posibilidad de usar la ayuda del servicio de reconocimiento óptico de caracteres para añadir sus gastos.

Se decidió, para facilitar el uso de la aplicación la eliminación de diferentes perfiles de uso para la aplicación, se considera que un móvil es usado por un solo usuario, y que añadir una funcionalidad de perfiles o cuentas, dificultaría su uso, ya que ralentizaría el añadir un gasto. Igualmente se decidió no añadir otras funcionalidades como son diferentes Divisas dentro de la aplicación.

Estos gastos estarán asignados a una categoría y a un lugar, el usuario también podrá gestionar las categorías y los lugares desde la aplicación. Se le ofrecerán al usuario varias categorías y lugares por defecto para que desde un principio pueda usar la aplicación sin preocuparse de crear categorías o lugares.

También podrá obtener los lugares de gasto mediante geo localización o mediante un mapa que se mostrara al usuario para que seleccione el lugar del gasto. Para esta posibilidad se usara el servicio de *Google Maps.*

Los gastos del usuario se mostraran inicialmente en una pantalla resumen, el usuario si lo desea podrá consultar los gastos desde otra pantalla de la aplicación. Desde esta pantalla el usuario podrá filtrar los gastos, mediante los distintos campos que forman un gasto.

Los gastos de la base de datos se podrán exportar en formato estándar CSV, para poder utilizarlo en diferentes programas de hojas de cálculo disponibles y que soportan este formato.

El usuario podrá consultar diferentes estadísticas y gráficos de los gastos que ha introducido, así el usuario tendrá de una forma gráfica información extra sobre sus hábitos de gasto. Podrá consultar graficas estadísticas de la distribución de sus gastos tanto en el tiempo, como por categorías y lugares. Mediante las preferencias de la aplicación el usuario podrá personificar la aplicación.

1.4 ESTRUCTURA DE LA MEMORIA DEL PROYECTO

La memoria del presente Proyecto Final de Carrera se ha dividido en 9 capítulos, en los cuales documentaremos los diferentes procesos realizados durante la realización del Proyecto.

Este es un pequeño resumen del contenido de los distintos capítulos de la memoria del Proyecto Final de Carrera:

- **Introducción:** En este capítulo de la memoria describiremos que es un Proyecto Final de Carrera, sus objetivos, motivación y una introducción a la funcionalidad final.

- **Evaluación Tecnológica:** En este capítulo analizaremos las diferentes tecnologías usadas en el Proyecto Final de Carrera, y las motivaciones para usarlas.
- **Planificación Inicial**: En esta parte de la memoria se explicara todo lo relacionado con la planificación inicial del Proyecto, y se analizara el coste temporal del mismo.
- **Análisis y especificación:** Se procederá a realizar la explicación del análisis de la Aplicación, y los requisitos que deberá cumplir.
- **Diseño**: En esta etapa se definirán los diferentes componentes del sistema.
- **Implementación**: Desarrollo del código de la aplicación.
- Planificación Final: Se analizara el coste temporal real del proyecto, y se comparara con el de la planificación Inicial.
- **Conclusiones:** Conclusiones finales alcanzadas, tanto personales como a nivel del proyecto, y análisis de posibles mejoras futuras.
- **Bibliografía:** Bibliografía y documentación utilizada durante la realización del proyecto.
- **Anexos**: Manual de uso de la aplicación.

2. EVALUACIÓN TECNOLÓGICA

2.1 ANDROID

El sistema operativo Android fue desarrollado inicialmente por la empresa Android Inc., después fue comprada por Google en el año 2005 debido del interés de esta por entrar en el mercado de los Smartphone. Android es una plataforma de código abierto distribuida bajo la licencia Apache 2.0

Poco después en el año 2007 se creó la Open Handset Alliance, que es una agrupación de fabricantes de móviles, chipsets, procesadores y la misma Google como directora de la misma. Se proporcionó el código fuente de la primera versión de Android, junto su SDK, y la primera versión del sistema operativo Android.

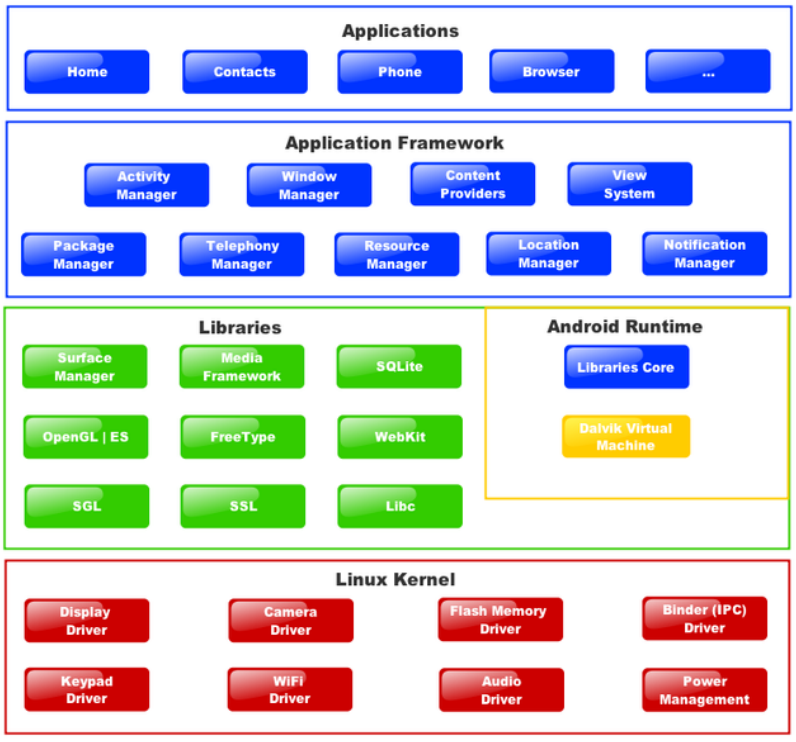
Para ver la importancia de Android podemos indicar que la Open Handset Alliance está formada por las siguientes empresas en la actualidad: Texas Instruments, Broadcom Corporation, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, Intel, LG, Marvell Technology Group, Motorola, T-Mobile, PacketVideo, ARM Holdings, Atheros Communications, Asustek, Garmin, Softbank, Sony Ericsson, Toshiba, Vodafone y ZTE.

Está basado en GNU/Linux, y la versión del Kernel de Linux utilizada en cada versión del sistema operativo varia. Así que se puede decir que el núcleo del sistema operativo está escrito en C.

La parte del interfaz de usuario de Android está desarrollada en Java. Android dispone de una avanzada máquina virtual de java llama Dalvik. La parte grafica del sistema operativo soporta OpenGL ES, variando la versión soportada de OpenGL dependiendo de la versión del sistema Android.

Como motor de base de datos usa una versión ligeramente modificada de SQLite. El sistema operativo también incluye un motor de navegación Html basado en webkit.

A continuación podemos ver un esquema de la estructura del sistema operativo Android:



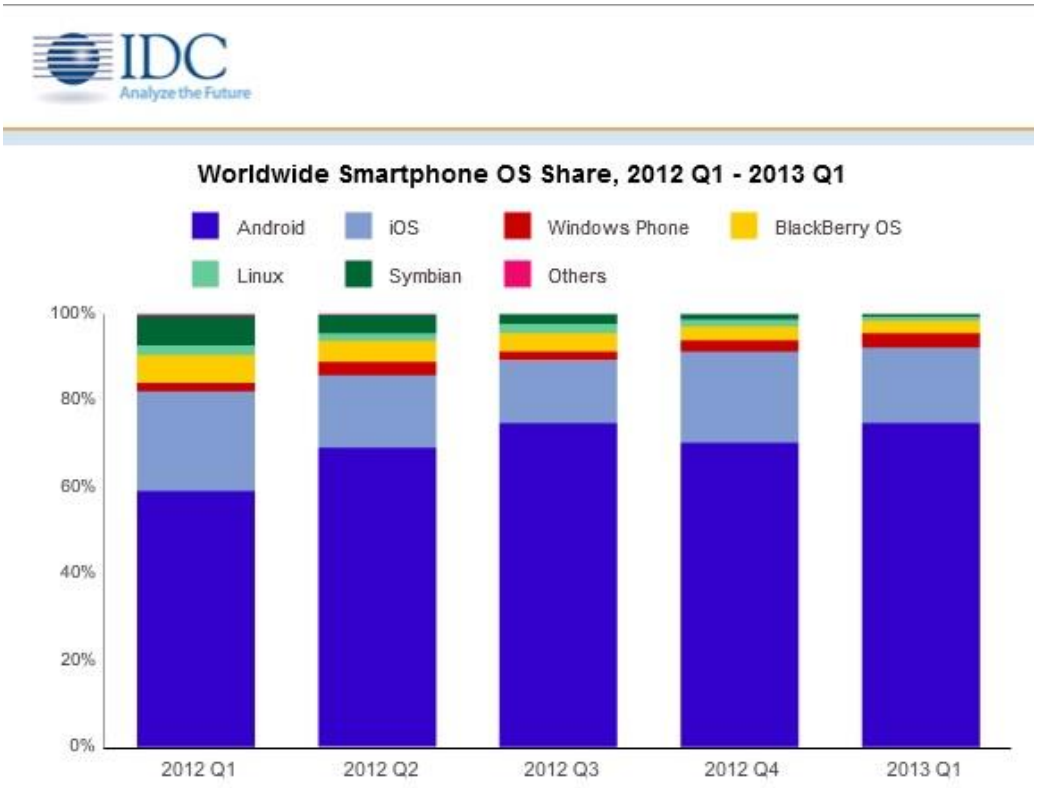
2.1.1. EVOLUCIÓN DE ANDROID EN EL MERCADO

Una de las razones para elegir Android como plataforma para desarrollar el proyecto ha sido su expansión en el mercado de Smartphone. Después de unos inicios modestos, ahora mismo Android se sitúa como el principal Sistema Operativo para Smartphone y Tablet.

Como podemos ver en la siguiente tabla, el crecimiento en el último año por ejemplo es del 26% del mercado, pasando de tener el 59% del mercado de Smartphone al 75%.

Operating System	1Q13 Shipment Volume	1Q13 Market Share	1Q12 Shipment Volume	1Q12 Market Share	Year over Year Change
Android	162.1	75.0%	90.3	59.1%	79.5%
iOS	37.4	17.3%	35.1	23.0%	6.6%
Windows Phone	7.0	3.2%	3.0	2.0%	133.3%
BlackBerry OS	6.3	2.9%	9.7	6.4%	-35.1%
Linux	2.1	1.0%	3.6	2.4%	-41.7%
Symbian	1.2	0.6%	10.4	6.8%	-88.5%
Others	0.1	0.0%	0.6	0.4%	-83.3%
Total	**216.2**	**100.0%**	**152.7**	**100.0%**	**41.6%**

Fuente: IDC Worldwide Quarterly Mobile Phone Tracker, May 2013.



2.1.2. VERSIONES DE ANDROID

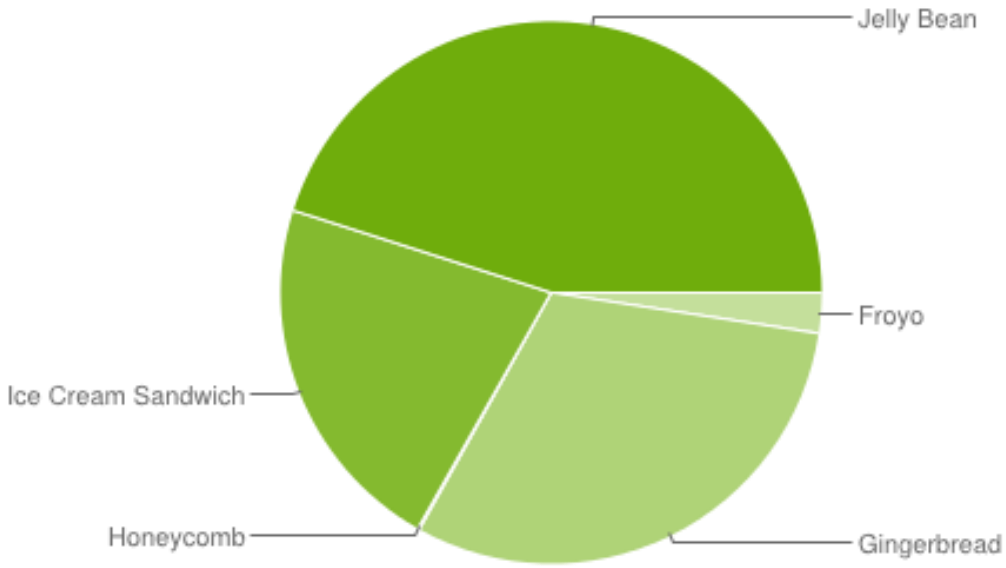
Una de las primeras decisiones que han sido necesarias tomar, es sobre que versión de la plataforma Android funcionaria la aplicación.

Android ha sufrido una rápida evolución, *Google* ha lanzado versiones nuevas del sistema operativo constantemente, esto ha creado una fragmentación en la plataforma, ya que hay funcionalidades que existen en una versión pero no en otra.

Con cada versión de Android, Google lanza un nuevo SDK, con nuevas Apis y con cambios en las Apis antiguas. En la actualidad existen las siguientes versiones de Android:

- Android 1.0 Apple Pie
- Android 1.1 Banana Bread
- Android 1.5 Cupcake
- Android 1.6 Donut
- Android 2.0/2.1 Eclair
- Android 2.2.x Froyo
- Android 2.3.x Gingerbread
- Android 3.x Honeycomb
- Android 4.0.x Ice Cream Sandwich
- Android 4.1 Jelly Bean
- Android 4.2 Jelly Bean
- Android 4.3 Jelly Bean

De la versión 1.0 a la versión 2.2 se consideran ya versiones obsoletas.
En la siguiente grafica podemos ver la distribución de las versiones en el mercado actual:



A continuación se adjunta una tabla de distribución de Versiones de Android:

VERSIÓN	NOMBRE	API	% DISTRIBUCIÓN
2.2	Froyo	8	2.4%
2.3	Gingerbread	10	30.7%
3.2	Honeycomb	13	0.1%
4.03	Ice Cream Sandwich	15	21.7%
4.1	Jelly Bean	16	36.6%
4.2	Jelly Bean	17	8.5%

Después de analizar estos datos, y ver que la evolución de esta distribución tiende a que las versiones inferiores de la versión 4 se reduzcan al mínimo, se tomó la decisión de que la aplicación se desarrollaría con la versión 3+ de Android en mente. Esto implica que el SDK mínimo utilizado sea la versión 11.

A partir de la versión 3 se produjeron los cambios más importantes en la plataforma Android, como son la inclusión de la Barra de Acción o los fragmentos que se detallaran más adelante.

2.1.3. COMPONENTES DE UNA APLICACIÓN ANDROID

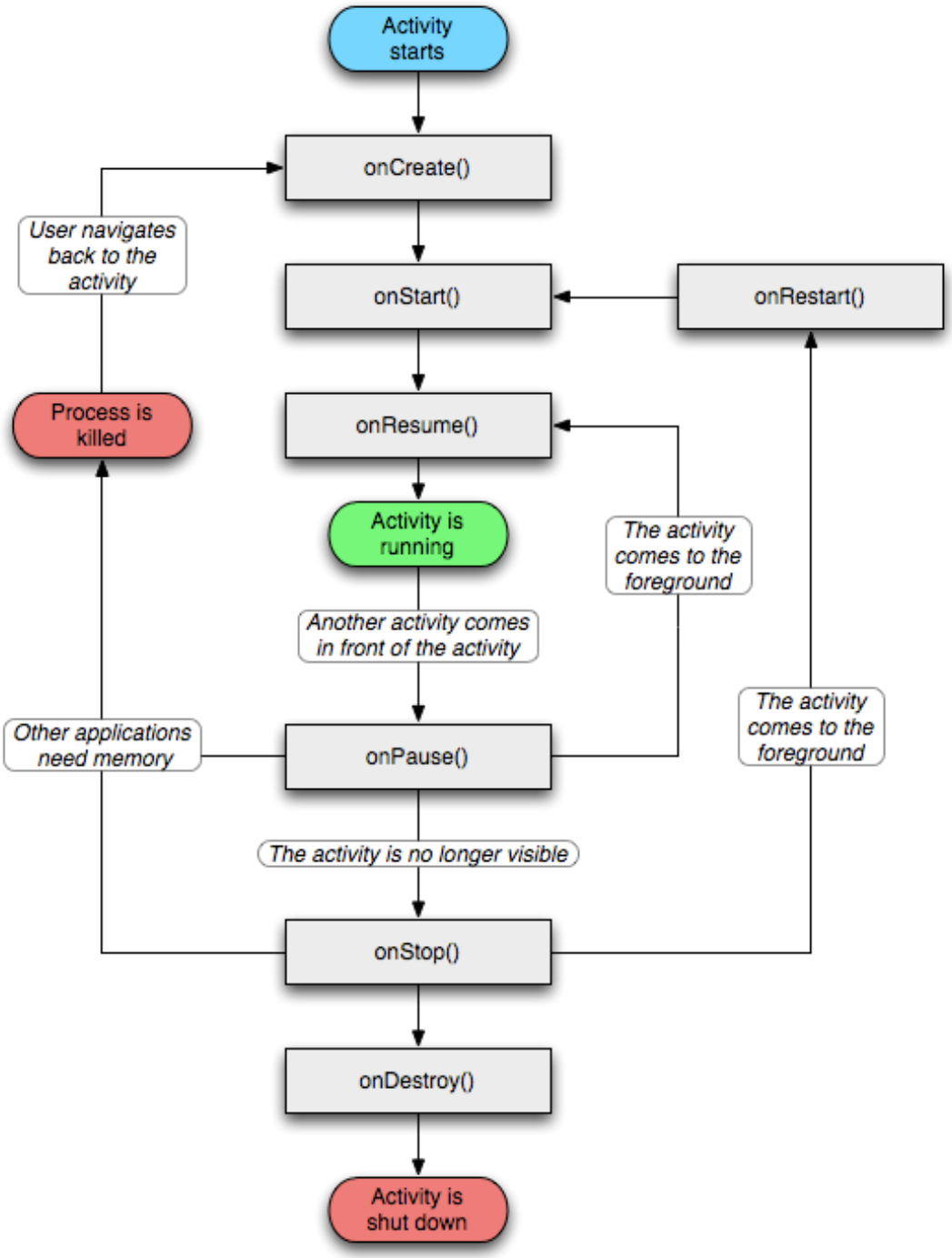
Una aplicación Android está compuesta de unos componentes. Una aplicación no necesita implementar todos estos componentes, pero usualmente a excepción de alguno de ellos, están implementados la mayoría.

- **View:** Las view son los elementos que forman la interfaz de usuario de la aplicación. Las views son definidas de forma estática mediante código XML, o mediante Java de forma dinámica. De la clase View descienden los componentes como pueden ser los botones (Button) o Spinner (selector).
- **Layout**: Son un conjunto de views agrupadas de una forma determinada. Más adelante veremos estas diferentes agrupaciones. Como las views se pueden definir de forma estática mediante código XML, o mediante Java de forma dinámica.
- **Activity:** Las aplicaciones Android están formadas de 1 o más Actividades. Estas actividades suelen corresponder a las diferentes pantallas o funcionalidades de la aplicación. Su principal función es crear el interfaz de usuario. Las diferentes actividades que forman una aplicación son independientes entre sí, pero se pueden relacionar entre ellas para pasar información.
- **Fragment:** Es una de las novedades en la versión 3 de Android. Son actividades pequeñas que pueden mostrar diferente información, y ser usadas de forma más dinámica que una Activity normal.
- **Service**: Es un proceso que se ejecuta en segundo plano, transparentemente al usuario. A diferencia de una Activity no tiene un interfaz de usuario.
- **Intent**: Es la forma que tiene Android de lanzar una Actividad propia de la aplicación, u otra externa del sistema operativo Android o instalada por el usuario.

2.1.4. CICLO DE VIDA DE UNA APLICACIÓN ANDROID

Una actividad de Android posee un ciclo de vida para realizar el proceso para el cual ha sido diseñada. El sistema operativo es el encargado de ejecutar o eliminar de la memoria la Aplicación.

El sistema operativo también controla la aplicación dependiendo de una serie de estados que puede tener una actividad.



El ciclo de vida de un **fragment** es prácticamente equivalente a la de una **activity**.

2.1.5. COMPONENTES DE LA INTERFAZ DE USUARIO DE ANDROID (WIDGETS)

Un view está formado por diferentes componentes o widgets. Los widgets disponibles en Android son más de 50, aquí hay un pequeño resumen de los más utilizados:

- Button
- TextView
- EditText
- ListView
- Checkbox
- Spinner
- ImageView

A todos estos widgets se les puede asignar diferentes eventos relacionas con la interacción del usuario con el mismo.

2.1.6. LAYOUTS DE UNA APLICACIÓN ANDROID

Un layout define como se mostraran en pantallas los diferentes componentes de las views. Hay distintos tipos de layouts:

- Linear Layout
- Relative Layout
- Table Layout
- Absolute Layout
- Frame Layout

2.2 JAVA

Java es un lenguaje de programación orientado a objetos diseñado por Sun Microsistems en los años 90. Su sintaxis deriva de otros lenguajes de programación como son C y C++.

Una de sus principales características es su facilidad para funcionar en múltiples plataformas, ya que sus aplicaciones son compiladas a bytecode y se ejecuta en una máquina virtual de Java. En el caso de Android esta máquina virtual se llama Dalvik.

La segunda es que facilita al programador la gestión de memoria, ya que implementa un recolector de basura, que se encarga de liberar memoria de los objetos que ya no están en uso por el programa.

2.3 XML

XML no es un lenguaje de programación sino un estándar de codificación de información. Es un lenguaje de marcas como el HTML que se utiliza para almacenar información de forma legible.

A diferencia de HTML, el XML no ha nacido como para uso exclusivo en internet, sino que se propone como estándar para compartir información de forma estructurada en diferentes plataformas.

Por ejemplo está siendo usado, por ejemplo está siendo usado en diferente software desde editores de texto a sistemas operativos como Android.

2.4 SQL

El lenguaje SQL es un lenguaje de acceso a Base de Datos relacionales. Este lenguaje permite consultar, modificar y añadir información a una Base de Datos de forma sencilla.

2.4.1. SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB)2 biblioteca escrita en C.

Es un proyecto de dominio público. SQLite a diferencia de otros sistemas de gestión de Bases de Datos, no funciona externamente, sino que se integra como parte del programa. Esto aumenta la velocidad de las operaciones con la BD.

El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como aplicación fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

SQLite forma parte por defecto de la distribución del sistema operativo Android.

2.5 RECONOCIMIENTO ÓPTICO DE CARACTERES.

El reconocimiento óptico de caracteres u OCR en inglés, es un proceso dirigido a la digitalización de textos, estos sistemas de OCR identifican automáticamente mediante una imagen los diferentes caracteres de un alfabeto, y luego los almacenan en algún formato de datos.

El proceso que realiza un sistema de reconocimiento óptico de caracteres es él a modo de resumen el siguiente:

> *Se debe partir de una imagen perfecta, se considera una imagen perfecta una imagen con solo dos niveles de grises, el reconocimiento de los caracteres de la imagen se realizara comparando los caracteres de la imagen con diferentes patrones o plantillas que contienen todos los diferentes caracteres del alfabeto.*

Posibles problemas para reconocer caracteres:

- El dispositivo que toma la imagen puede variar los niveles de grises de la imagen, modificando la imagen original.
- La resolución puede no ser suficiente, lo que puede provocar que se añada ruido a la imagen.
- La distancia que separa a los distintos caracteres también es un problema, ya que es una de las principales fuente de problemas para reconocer caracteres.
- Igualmente si varios de estos caracteres están conectados entre sí puede provocar problemas para reconocer los 2 caracteres.
- Cualquier información que no sea caracteres, provocara ruido y que la imagen no se pueda reconocer de forma correcta.

En el Anexo 2 de la memoria se puede consultar información sobre la tecnología de Reconocimiento Óptico de Caracteres.

2.5.1 RECONOCIMIENTO ÓPTICO DE CARACTERES EN ANDROID.

Para este proyecto he tenido que estudiar las diferentes posibilidades para utilizar reconocimiento óptico de caracteres en Android.

Estas son las diferentes opciones que existen en Android:

- **ABBYY:** http://www.abbyyeu.com/

  La empresa ABBYY tiene una variedad de productos relacionados con OCR, entre ellos *"Mobile OCR Engine"*, es un producto de gran calidad con muy buenos resultados según los análisis de diferentes.

  Tiene un gran problema que la elimina completamente de las posibilidades, que es su alto precio, 1000$ por el motor básico de OCR, y un mínimo del 20% del dinero obtenido por las ventas de la aplicación.

- **GOCR / JOCR**

  Es una alternativa de OCR en código libre, eso hacía que fuera una atractiva alternativa, pero la falta de documentación, y la dificultad de utilizarla Android finalmente causó que fuera descartada.

- **TESSREACT-OCR**

  Es una librería de código libre con licencia apache 2.0

Fue creada por HP en los años 80 y en la actualidad está siendo mantenido por Google. Es multiplataforma, y es una de las librerías OCR con mejor reconocimiento y compatibilidad con diferentes formatos gráficos

Finalmente la elección fue la librería tess-two que es una adaptación de tessreact-ocr para Android de esta librería. Pese a alguna limitación como los límites de memoria que se pueden usar en Android por una aplicación, es la alternativa más eficaz y el ser de código libre la hace la más adecuada de todas las analizadas.

Más información sobre tess-two en https://github.com/rmtheis/tess-two

2.6 LIBRERÍAS GENERADORAS DE GRÁFICAS ESTADÍSTICAS

Una de las funcionalidades de la Aplicación es mostrar graficas estadísticas de la información guardada en la Base de Datos, para esto hemos de utilizar alguna de las librerías que generan este tipo de gráficas para Android.

Después de un análisis de las posibles alternativas, se puede decir que es un apartado que Android debería mejorar, ya que la mayoría de estas librerías sufren bastantes deficiencias o limitaciones.

Estas son las diferentes alternativas de librerías para graficas estadísticas:

- **ANDROIDPLOT**
  Es una librería de código libre con licencia apache 2.0, el problema de esta librería es la falta de documentación, e información, cosa que dificulto su posible implementación.
- **GOOGLE CHARTS**
  No es realmente una librería, esta alternativa requería conexión a internet de forma constante, y mostrar los resultados en una página web. Es muy potente y flexible pero los requisitos necesarios hicieron que descartara su uso.
- **CHARTDROID**
  Tampoco es una librería, sino una aplicación externa que se usa para mostrar gráficos. Esto provocaba que el usuario deba instalar este programa aparte de la aplicación propia en sí, debido a esto fue descartada.
- **ACHARTENGINE**
  Es una librería de código libre con licencia apache 2.0, esta librería tiene una seria de limitaciones como que gráficamente no destaca tanto como por ejemplo Androidplot, pero su extensa documentación y ejemplos permiten una fácil implementación.

La librería elegida finalmente fue ***Achartengine,*** pese a gráficamente no ser tan atractiva, es la alternativa más fácil de utilizar, y no requiere internet ni utilizar aplicaciones externas.

2.7 OTRAS LIBRERÍAS / APIS UTILIZADAS

- **LIBRERÍA JTAR** (https://code.google.com/p/jtar/)
  Los archivos de datos utilizados por la librería que da soporte a la funcionalidad de OCR, se encuentran en formato "TAR.GZ".
  En la plataforma Android se tiene soporte nativo para el formato GZ (GNU Zipped Archive file), pero para el formato TAR no existe ese soporte directo. Para desempaquetar el fichero TAR se ha utilizado la librería JTAR.

Las librerías / APIS de Google utilizadas en este proyecto han sido las siguientes:

- **Android.support.v4.app**

Es una librería que permite el uso de distintas funcionalidades en diferentes versiones de Android.

Esta librería está siendo actualizada constantemente por google para ofrecer nuevas funcionalidades sin tener que actualizar a una nueva versión del sistema operativo Android.

La principal funcionalidad utilizada ha sido la navegación lateral de la aplicación, y el soporte de fragmentos en diferentes versiones de Android.

- **Google play services lib**

Esta librería permite usar diferentes servicios de Google como es acceder a Google Maps desde la aplicación.

En concreto para esta aplicación se ha utilizado Google Maps Android API v2, que se accede mediante esta librería.

3. PLANIFICACIÓN INICIAL

Para planificar el proyecto, este se ha dividido en las diferentes tareas necesarias para realizar el mismo. Las tareas necesarias para completar el proyecto son las siguientes:

- **Organización y planificación del proyecto:** Realización de la planificación inicial del proyecto y organización del mismo.
- **Aprendizaje y documentación de las tecnologías utilizadas:** durante esta tarea nos familiarizaremos, y documentaremos las diferentes tecnologías utilizadas en este proyecto. Esta tarea se realizara en paralelo con el diseño, y análisis de la aplicación.
- **Análisis y diseño de la aplicación:** En esta fase analizaremos los objetivos, y definiremos el problema que tenemos que resolver, en este caso realizar una aplicación para gestionar gastos. Así como los pasos para resolver el mismo.
- **Implementación:** Con las tareas anteriores completadas, pasaremos a resolver el problema creando la aplicación software definida anteriormente.
- **Test y pruebas:** Durante esta fase se probara la aplicación en un entorno real para poder dar con posibles problemas, y mejorar la aplicación.
- **Correcciones:** En paralelo y como respuesta a la fase anterior, realizaremos las correcciones necesarias en el software para mejorar su funcionamiento.
- **Realización de la memoria:** Se realizara la memoria del proyecto.

3.1 COSTE TEMPORAL

TAREA	ANALISTA (días)	PROGRAMADOR (días)
Organización y planificación del proyecto	2	
Análisis de la aplicación	7	
Diseño de la aplicación	15	
Implementación		35
Test y pruebas		7
Correcciones		7
Realización de la memoria	7	
TOTAL	**31**	**49**
TOTAL DÍAS	**80**	
Meses estimados	**3.63 Meses**	
Horas estimadas	**640 Horas**	

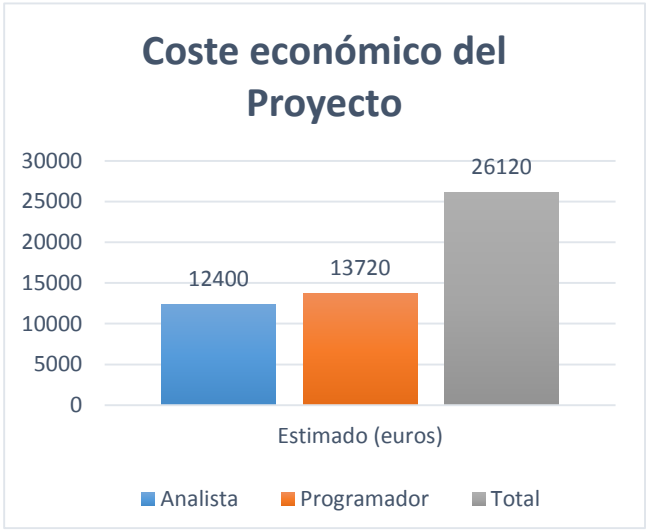
Para realizar el cálculo de meses estimado se han tomado como referencia que un día corresponde a 8 horas de trabajo, y que un mes tiene 22 días laborables.

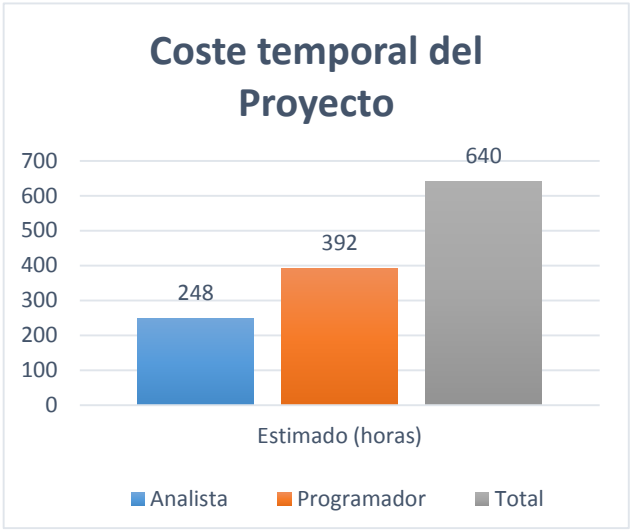
El cálculo estimado en meses es de 5.5 Meses, pero debido a que varias de las tareas se hacen en paralelo podemos ver en el Diagrama de GANTT que el tiempo estimado es menor.

3.2 COSTE ECONÓMICO

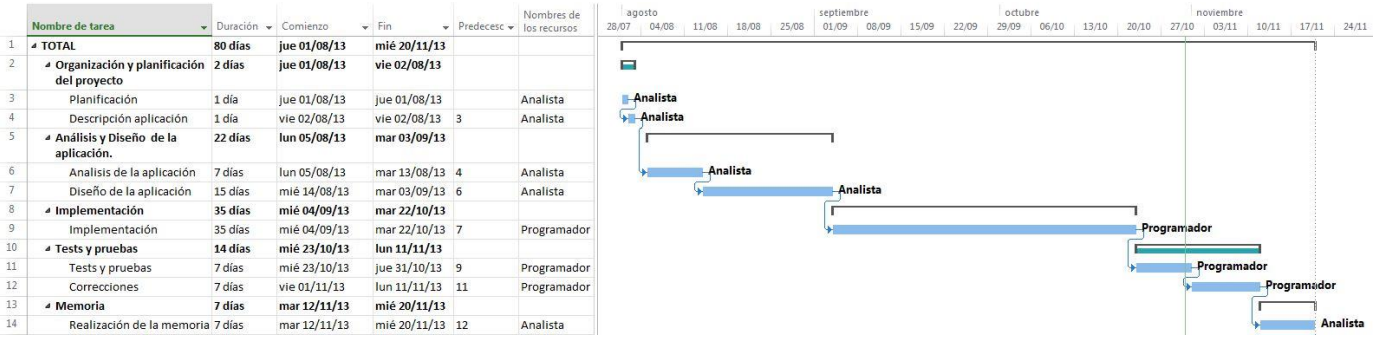
Si consideramos el coste por hora de un programador es de 35 euros, y el de un analista de 50 euros por hora, dentro de este coste por hora se incluyen todos los gastos asociados al realizar esta actividad y no solo el sueldo del trabajador, nos daría como resultado un coste de:

	ANALISTA	**PROGRAMADOR**
Horas	248 Horas	392 Horas
Coste	12400 Euros	13720 Euros
COSTE TOTAL	**26.120 EUROS**	





3.3 DIAGRAMA DE GANTT



	Nombre de tarea	Duración	Comienzo	Fin	Predecesc	Nombres de los recursos
1	**TOTAL**	**80 días**	**jue 01/08/13**	**mié 20/11/13**		
2	**Organización y planificación del proyecto**	**2 días**	**jue 01/08/13**	**vie 02/08/13**		
3	Planificación	1 día	jue 01/08/13	jue 01/08/13		Analista
4	Descripción aplicación	1 día	vie 02/08/13	vie 02/08/13	3	Analista
5	**Análisis y Diseño de la aplicación.**	**22 días**	**lun 05/08/13**	**mar 03/09/13**		
6	Analisis de la aplicación	7 días	lun 05/08/13	mar 13/08/13	4	Analista
7	Diseño de la aplicación	15 días	mié 14/08/13	mar 03/09/13	6	Analista
8	**Implementación**	**35 días**	**mié 04/09/13**	**mar 22/10/13**		
9	Implementación	35 días	mié 04/09/13	mar 22/10/13	7	Programador
10	**Tests y pruebas**	**14 días**	**mié 23/10/13**	**lun 11/11/13**		
11	Tests y pruebas	7 días	mié 23/10/13	jue 31/10/13	9	Programador
12	Correcciones	7 días	vie 01/11/13	lun 11/11/13	11	Programador
13	**Memoria**	**7 días**	**mar 12/11/13**	**mié 20/11/13**		
14	Realización de la memoria	7 días	mar 12/11/13	mié 20/11/13	12	Analista

- **Tareas:**

	Nombre de tarea	Duración	Comienzo	Fin	Predecesc	Nombres de los recursos
1	**TOTAL**	**80 días**	**jue 01/08/13**	**mié 20/11/13**		
2	**Organización y planificación del proyecto**	**2 días**	**jue 01/08/13**	**vie 02/08/13**		
3	Planificación	1 día	jue 01/08/13	jue 01/08/13		Analista
4	Descripción aplicación	1 día	vie 02/08/13	vie 02/08/13	3	Analista
5	**Análisis y Diseño de la aplicación.**	**22 días**	**lun 05/08/13**	**mar 03/09/13**		
6	Analisis de la aplicación	7 días	lun 05/08/13	mar 13/08/13	4	Analista
7	Diseño de la aplicación	15 días	mié 14/08/13	mar 03/09/13	6	Analista
8	**Implementación**	**35 días**	**mié 04/09/13**	**mar 22/10/13**		
9	Implementación	35 días	mié 04/09/13	mar 22/10/13	7	Programador
10	**Tests y pruebas**	**14 días**	**mié 23/10/13**	**lun 11/11/13**		
11	Tests y pruebas	7 días	mié 23/10/13	jue 31/10/13	9	Programador
12	Correcciones	7 días	vie 01/11/13	lun 11/11/13	11	Programador
13	**Memoria**	**7 días**	**mar 12/11/13**	**mié 20/11/13**		
14	Realización de la memoria	7 días	mar 12/11/13	mié 20/11/13	12	Analista

4. ANÁLISIS Y ESPECIFICACIÓN

4.1 ESPECIFICACIÓN DE REQUERIMIENTOS

4.1.1 REQUERIMIENTOS FUNCIONALES

Las funcionalidades requeridas en esta aplicación se pueden dividir en varias categorías. Desde esta aplicación deberá poderse realizar lo siguiente:

- Gestionar los Gastos del Usuario.
- Gestionar las diferentes categorías de los Gastos del Usuario
- Gestionar los diferentes lugares de los Gastos del Usuario
- Proporcionar listados de los Gastos del Usuario
- Proporcionar listados de las categorías de los Gastos del Usuario
- Proporcionar listados de los lugares de los Gastos del Usuario
- Generar y consultar estadísticas de los Gastos del Usuario.
- Gestionar las Preferencias de la Aplicación.
- Exportar la información de la aplicación.
- La aplicación solo gestionara los gastos de un usuario, o única cuenta, no se quiere dificultar el uso de la aplicación al usuario añadiendo diferentes cuentas que solo complicarían el uso de la aplicación.

A continuación, definiremos los diferentes requerimientos de forma más detallada.

• **GASTOS:**

R1.1. Añadir un nuevo Gasto por teclado.

- Se podrá añadir un nuevo Gasto a la Base de Datos mediante el teclado.
- Se podrá acceder a Añadir una nueva Categoría y un nuevo Gasto desde este proceso. Así, se podrá facilitar el uso de la aplicación al usuario.

R1.2. Añadir un nuevo Gasto por OCR.

- Se podrá añadir un nuevo Gasto a la Base de Datos mediante reconocimiento óptico de caracteres.

R1.3. Modificar un Gasto.

- Se podrá modificar un Gasto de la Base de Datos.

R1.4. Eliminar un Gasto.

- Se podrá eliminar un Gasto de la Base de Datos

R1.5. Consultar un Gasto.

- Se podrá consultar la información del Gasto de la Base de Datos.

R1.6. Generar listados de Gastos.

- Se podrá generar un listado de Gastos de la Base de Datos.
- Desde este listado de Gastos se podrá acceder a modificar el Gasto en la Base de Datos.
- Se podrán ordenar estos listados por los diferentes campos de un Gasto.
- Se podrá filtrar estos listados de Gastos por Categoría de Gasto y por Lugar de Gasto.

• **CATEGORÍAS:**

R2.1. Añadir una nueva Categoría de Gasto.

- Se podrá añadir una nueva Categoría de Gasto a la Base de Datos.

R2.2. Modificar una Categoría de Gasto.

- Se podrá modificar una Categoría de Gasto de la Base de Datos.

R2.3. Eliminar una Categoría de Gasto.

- Se podrá eliminar una Categoría de Gasto de la Base de Datos.
- Solo se podrá eliminar una Categoría de Gasto de la Base de Datos sino está en uso por un Gasto.

R2.4. Consultar una Categoría de Gasto.

- Se podrá consultar la información de la Categoría de Gasto de la Base de Datos.

R2.5. Generar listados de Categorías de Gasto.

- Se podrá generar un listado de Categorías de Gasto de la Base de Datos.
- Desde este listado de Gastos se podrá acceder a modificar la Categoría de Gasto en la Base de Datos.
- Se podrán ordenar estos listados por los diferentes campos de una Categoría de Gasto.

• **LUGARES:**

R3.1. Añadir un nuevo Lugar de Gasto Manualmente.

- Se podrá añadir un nuevo Lugar de Gasto a la Base de Datos.

R3.2. Añadir un nuevo Lugar de Gasto por geo localización

- Se podrá añadir un nuevo Lugar de Gasto a la Base de Datos usando el servicio de Geo localización de Android.

R3.2. Añadir un nuevo Lugar de Gasto por posición en un mapa

- Se podrá añadir un nuevo Lugar de Gasto a la Base de Datos mostrando al usuario un Mapa, y que pueda seleccionar el lugar del gasto.

R3.3. Modificar un nuevo Lugar de Gasto.

- Se podrá modificar un nuevo Lugar de Gasto de la Base de Datos.

R3.4. Eliminar un nuevo Lugar de Gasto.

- Se podrá eliminar un nuevo Lugar de Gasto de la Base de Datos.
- Solo se podrá eliminar un nuevo Lugar de Gasto de la Base de Datos sino está en uso por un Gasto.

R3.5. Consultar un Lugar de Gasto.

- Se podrá consultar la información del Lugar de Gasto de la Base de Datos.

R3.6. Generar listados de Lugares de Gasto.

- Se podrá generar un listado de Lugares de Gasto de la Base de Datos.
- Desde este listado de Gastos se podrá acceder a modificar el lugar de Gasto en la Base de Datos.
- Se podrán ordenar estos listados por los diferentes campos de un Lugar de Gasto.

• **ESTADÍSTICAS:**

R4.1. Generar Estadísticas de los Gastos de Usuario.

- Se podrá generar Estadísticas de los distintos Gastos de la Base de Datos.
- Las Estadísticas podrán mostrar la distribución de los Gastos de la Base de Datos por fecha, por categoría, y por lugar.
- Se podrá mostrar el Gasto medio, máximo, mínimo y total por un rango de fechas.

• **RESUMEN:**

R5.1. Generar resumen mensual de los Gastos de Usuario.

- Se podrá generar un resumen mensual de los Gastos del Usuario.
- En este resumen se mostraran la cantidad de últimos gastos que el usuario elija en las preferencias de la Aplicación.
- Se mostrara información de los Gastos mensuales y de los gastos del día Actual, estos gastos se mostraran en formato texto, y de forma gráfica.
- El usuario podrá establecer un tope de Gasto Mensual, en el resumen se mostrara una comparación de los Gastos mensuales, y diarios con los del presupuesto definido por el usuario.

• **PREFERENCIAS:**

R6.1. Mostrar preferencias de la Aplicación.

- Se podrá mostrar una pantalla de preferencias al Usuario.

- Se mostrara cuantos Gastos se verán en el Resumen mensual de últimos Gastos.
- Se mostrara la cantidad destinada a presupuesto que será utilizada en el Resumen mensual de últimos Gastos.

R6.2 Modificar preferencias de la Aplicación.

- El usuario podrá modificar las preferencias de la aplicación.
- El usuario podrá elegir cuantos Gastos se verán en el Resumen mensual de últimos Gastos.
- El usuario podrá elegir la cantidad destinada a presupuesto que será utilizada en el Resumen mensual de últimos Gastos.

R6.3 Guardar preferencias de la Aplicación

- El usuario podrá guardar las preferencias de la aplicación de forma permanente.

• **EXPORTAR:**

R7.1 Exportar Base de Datos a CSV.

- El usuario podrá exportar la información de sus gastos, categorías de gasto y Lugares de gasto en formato CSV, para poder usar esta información como considere adecuado.

4.1.2 REQUERIMIENTOS NO FUNCIONALES

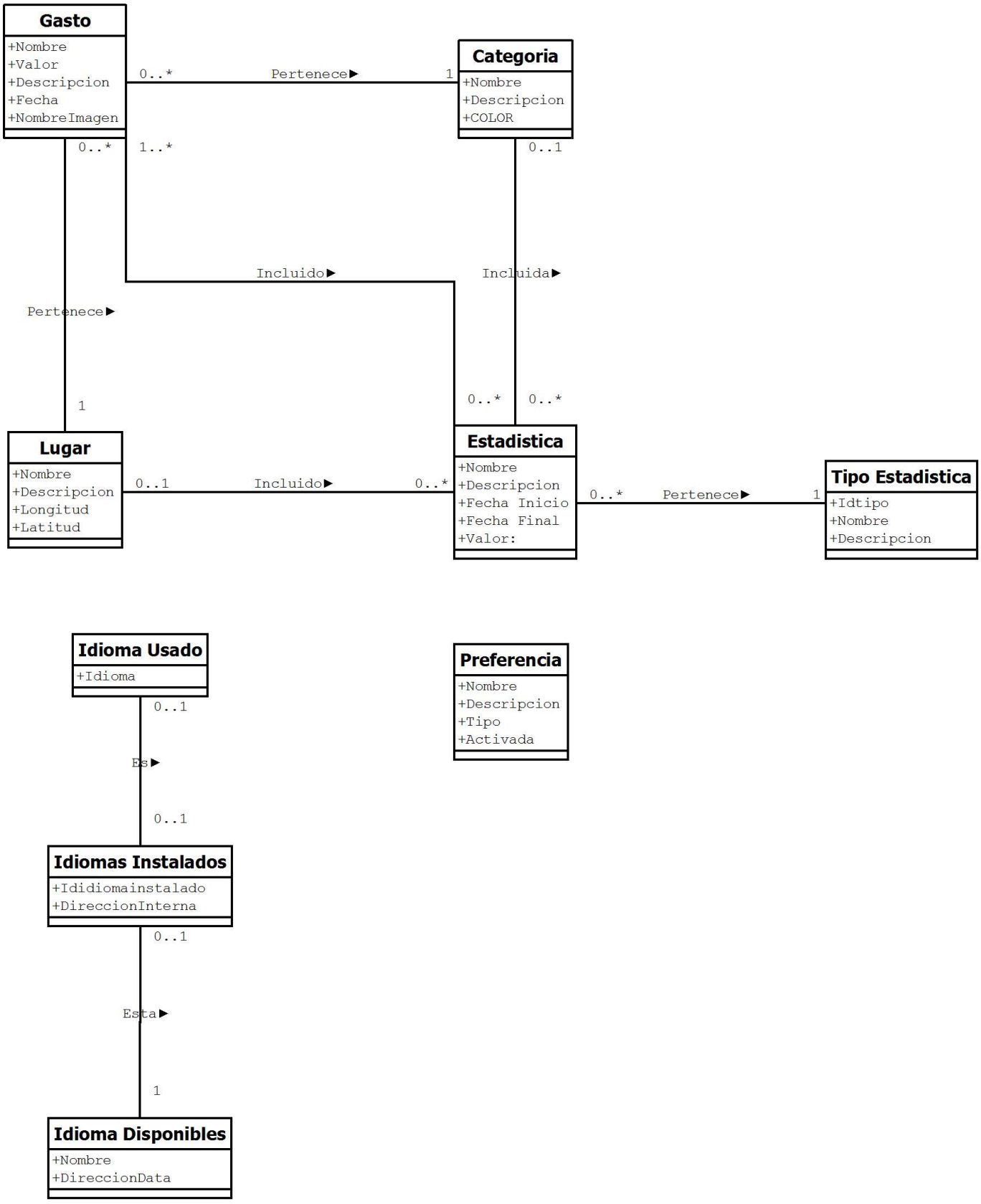
- La aplicación será multilingüe, y se podrá utilizar en Ingles, castellano y Catalán.
- Facilidad de Uso, la aplicación tendrá una interfaz clara e intuitiva.
- Software Libre: para reducir costes, el software utilizado serán aplicaciones libres en su gran mayoría.

4.1.3 REQUERIMIENTOS DEL SISTEMA

- Se requiere un móvil con versión de Android igual o superior a la versión 4.

4.2 MODELO CONCEPTUAL

UML



4.3 CASOS DE USO

Los diagramas de casos de uso muestran la relación entre los actores del sistema y los casos de uso de la aplicación.

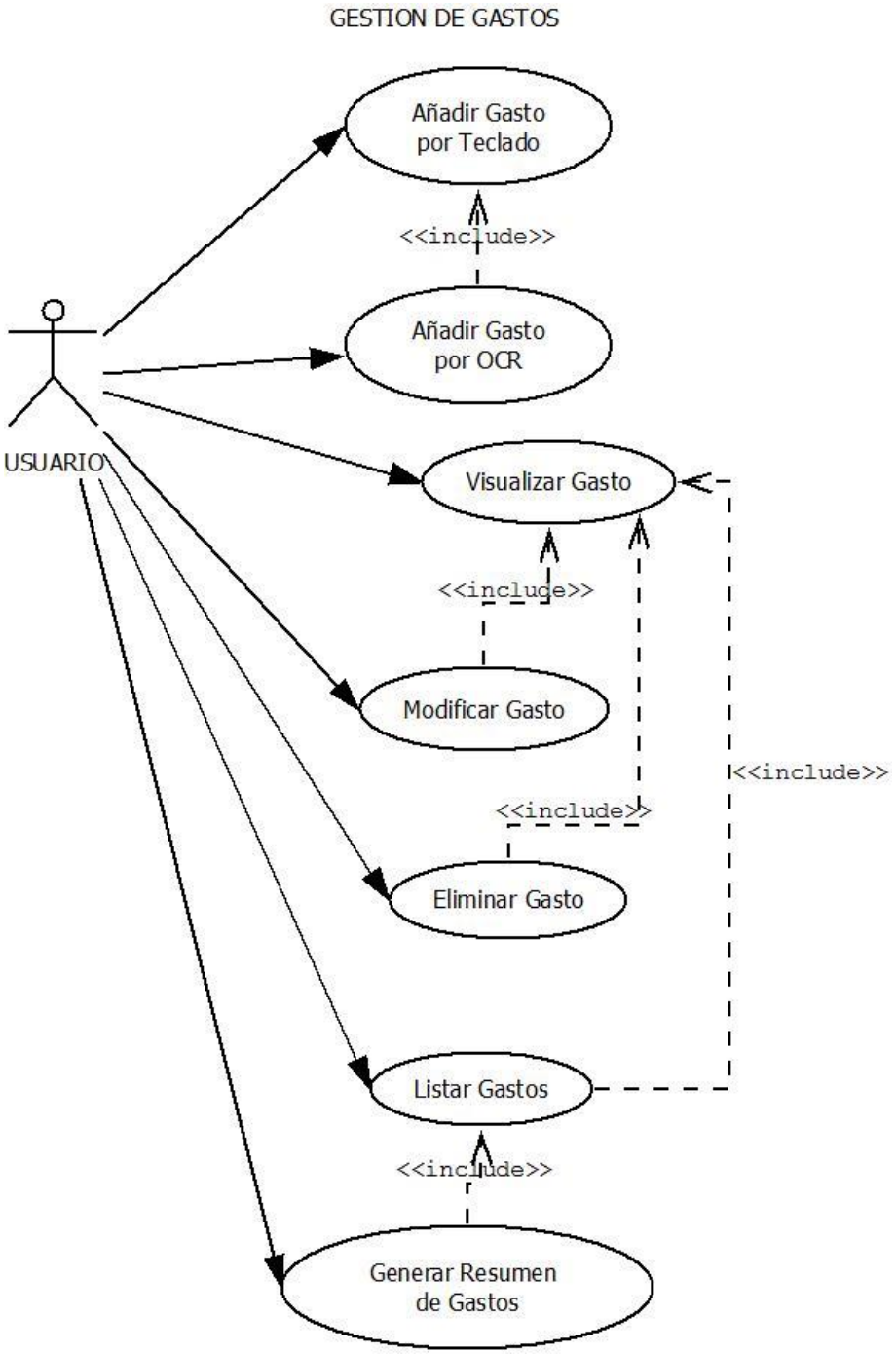
En el caso de esta aplicación solo existe un Actor, ya que la aplicación está concebida para que el usuario pueda fácilmente realizar las tareas de usuario y las mínimas tares de administración.

Actor de los casos de Usos detallados a continuación es Usuario.

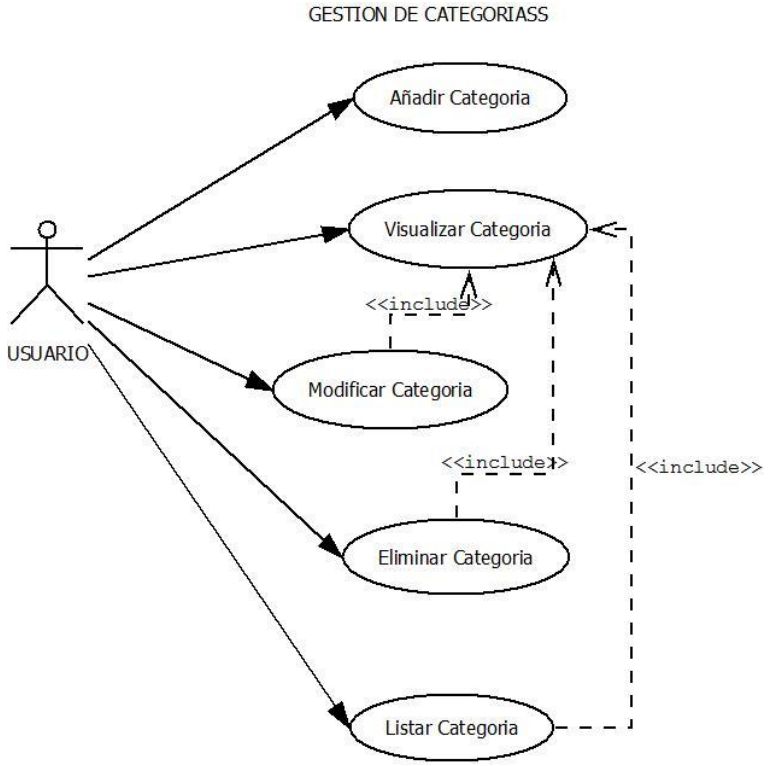
- **Include:** Indica que en la relación de los casos de usos el origen de la flecha incluye parte de los pasos del caso del final de la flecha, para simplificar este caso de uso y no repetir tareas se utiliza el *include.*
- **Extends:** Indica que el caso de uso apuntando por la flecha puede incluir el otro caso de uso.

4.3.1 DIAGRAMAS DE CASOS DE USO

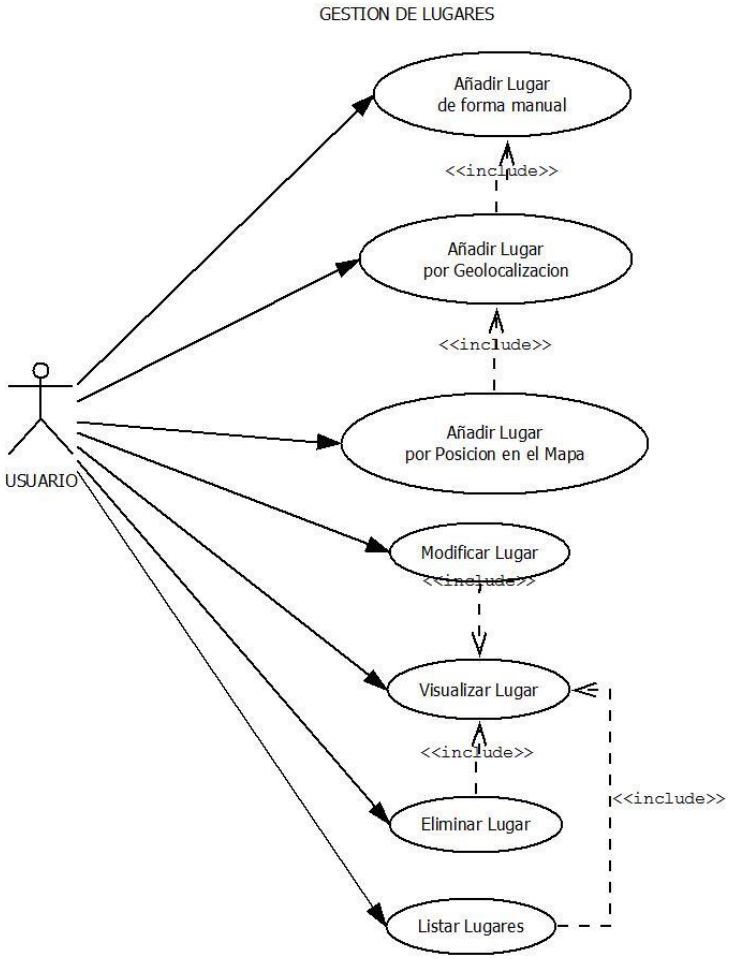
4.3.1.1 CASOS DE USO DE GASTOS



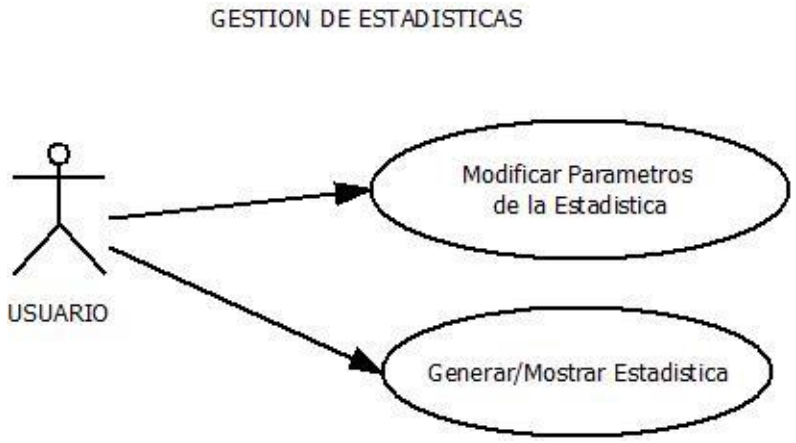
4.3.1.2 CASOS DE USO DE CATEGORÍAS



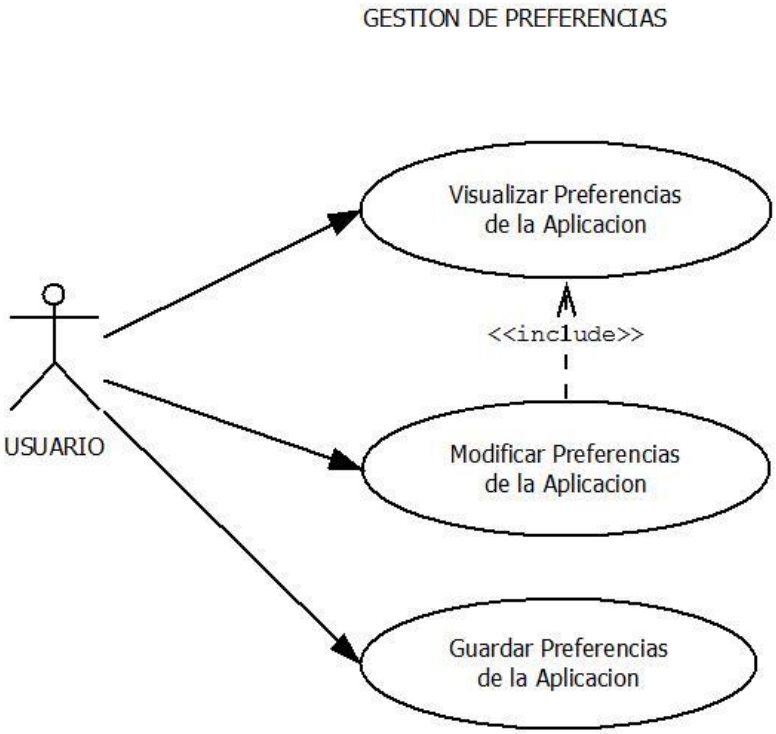
4.3.1.3 CASOS DE USO DE LUGARES



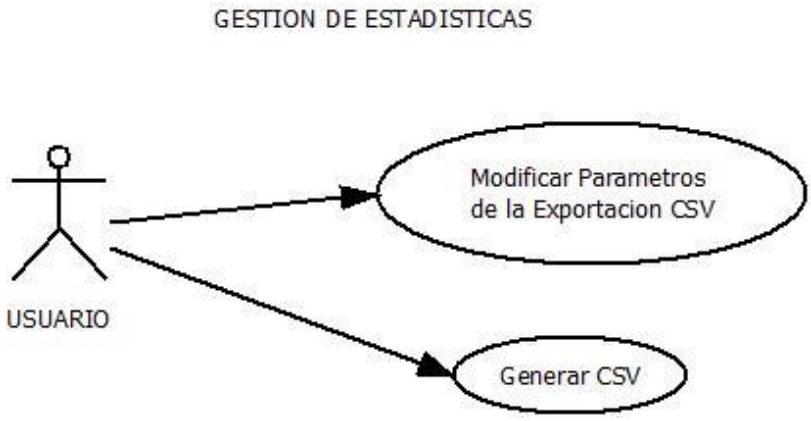
4.3.1.4 CASOS DE USO DE ESTADÍSTICAS.



4.3.1.5 CASOS DE USO DE PREFERENCIAS



4.3.1.6 CASOS DE USO DE EXPORTAR CSV



4.3.2 DESCRIPCIÓN DE CASOS DE USO

En este apartado se describe en formato tabla cada uno de los casos de uso del sistema.

4.3.2.1 AÑADIR GASTO POR TECLADO

CASO DE USO	AÑADIR GASTO POR TECLADO
Actores	Usuario
Propósito	Dar de alta un gasto en la Base de Datos
Resumen	El usuario accede a la opción de añadir gasto, rellena el formulario con los datos del gasto. La aplicación almacena el gasto en la Base de Datos
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de añadir gasto del menú.	
	2 La aplicación muestra el formulario de añadir gasto por teclado.
3 El usuario rellena el formulario, y aprieta el botón guardar	
	4 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
5 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 3.	

4.3.2.2 AÑADIR GASTO POR RECONOCIMIENTO ÓPTICO DE CARACTERES

CASO DE USO	AÑADIR GASTO POR RECONOCIMIENTO ÓPTICO DE CARACTERES
Actores	Usuario
Propósito	Dar de alta un gasto en la Base de Datos
Resumen	El usuario accede a la opción de añadir gasto por reconocimiento óptico de caracteres, toma una foto, rellena el formulario con los datos del gasto. La aplicación almacena el gasto en la Base de Datos
Tipo	Primario y esencial.
CURSO TÍPICO	

Acción del actor	Respuesta del sistema
1 El usuario accede a la opción de añadir gasto por reconocimiento óptico de caracteres del menú.	
	2 La aplicación muestra el formulario de añadir gasto por reconocimiento óptico de caracteres.
3 El usuario toma una foto, rellena el formulario, y aprieta el botón guardar	
	4 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
5 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 3.	

4.3.2.3 VISUALIZAR DATOS DE UN GASTO

CASO DE USO	**VISUALIZAR DATOS DE UN GASTO**
Actores	Usuario
Propósito	Mostrar todos los datos de un Gasto de la Base de Datos
Resumen	El usuario accede a la opción de lista de gastos, el usuario selecciona un gasto, y la aplicación le muestra los datos del Gasto.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de gastos del menú.	
	2 La aplicación muestra el listado de gastos.
3 El usuario selección uno de los gastos del listado de gastos.	
	4 La aplicación muestra el formulario de datos del gasto.
CURSOS ALTERNATIVOS	
No hay cursos alternativos.	

4.3.2.4 MODIFICAR DATOS DE UN GASTO

CASO DE USO	MODIFICAR DATOS DE UN GASTO
Actores	Usuario
Propósito	Modificar los datos de un Gasto de la Base de Datos
Resumen	El usuario accede a la opción de lista de gastos, el usuario selecciona un gasto, y la aplicación le muestra los datos del Gasto. El usuario modifica los datos del gasto, y la aplicación guarda los datos modificados en la Base de Datos.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de gastos del menú.	
	2 La aplicación muestra el listado de gastos.
3 El usuario selección uno de los gastos del listado de gastos.	
	4 La aplicación muestra el formulario de modificar datos de gasto.
5 El usuario modifica los datos del formulario, y aprieta el botón guardar	
	6 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
7 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 5.	

4.3.2.5 ELIMINAR GASTO DE LA BASE DE DATOS

CASO DE USO	ELIMINAR GASTO DE LA BASE DE DATOS
Actores	Usuario
Propósito	Eliminar un gasto definitivamente de la base de datos.
Resumen	El usuario accede a la opción de lista de gastos, el usuario selecciona un gasto, y la aplicación le muestra los datos del Gasto. El usuario selecciona la opción de eliminar el gasto, y la aplicación elimina el gasto de Base de Datos.
Tipo	Primario y esencial.
CURSO TÍPICO	

Acción del actor	Respuesta del sistema
1 El usuario accede a la opción de lista de gastos del menú.	
	2 La aplicación muestra el listado de gastos.
3 El usuario selección uno de los gastos del listado de gastos.	
	4 La aplicación muestra el formulario de modificar datos de gasto.
5 El usuario aprieta el botón eliminar gasto del formulario.	
	6 La aplicación elimina el gasto de la base de datos
CURSOS ALTERNATIVOS	
No hay cursos alternativos.	

4.3.2.6 LISTAR GASTOS DE LA BASE DE DATOS.

CASO DE USO	**LISTAR GASTOS DE LA BASE DE DATOS**
Actores	Usuario
Propósito	Listar todos los gastos de la base de datos.
Resumen	El usuario accede a la opción de lista de gastos, la aplicación le muestra un listado de los Gastos de la base de datos.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de gastos del menú.	
	2 La aplicación muestra el listado de gastos.
CURSOS ALTERNATIVOS	
No hay cursos alternativos.	

4.3.2.7 MOSTRAR RESUMEN DE GASTOS DE LA BASE DE DATOS

CASO DE USO	MOSTRAR RESUMEN DE GASTOS DE LA BASE DE DATOS
Actores	Usuario
Propósito	Listar un resumen de los últimos gastos de la base de datos.
Resumen	El usuario accede a la opción de resumen de gastos, la aplicación muestra el resumen de Gastos de la BD
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de resumen de gastos del menú.	
	2 La aplicación muestra el resumen de gastos.
CURSOS ALTERNATIVOS	
No hay cursos alternativos.	

4.3.2.8 AÑADIR CATEGORÍA DE GASTO A LA BASE DE DATOS.

CASO DE USO	AÑADIR CATEGORÍA DE GASTO A LA BASE DE DATOS
Actores	Usuario
Propósito	Dar de alta una categoría de Gasto en la Base de Datos
Resumen	El usuario accede a la opción de añadir categoría, rellena el formulario con los datos del gasto. La aplicación almacena la categoría en la Base de Datos
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de añadir categoría del menú.	
	2 La aplicación muestra el formulario de añadir categoría.
3 El usuario rellena el formulario, y aprieta el botón guardar	

	4 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
5 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 3.	

4.3.2.9 VISUALIZAR DATOS DE UNA CATEGORÍA DE GASTO

CASO DE USO	**VISUALIZAR DATOS DE UNA CATEGORÍA DE GASTO**
Actores	Usuario
Propósito	Mostrar todos los datos de una Categoría de Gasto de la Base de Datos
Resumen	El usuario accede a la opción de lista de categorías, el usuario selecciona una categoría, y la aplicación le muestra los datos de la categoría.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de categorías del menú.	
	2 La aplicación muestra el listado de categorías.
3 El usuario selección una de las categorías del listado de categorías.	
	4 La aplicación muestra el formulario de datos de la categoría.
CURSOS ALTERNATIVOS	
5 No existe ninguna categoría en la base de datos, al usuario se le muestra un mensaje avisándole de la situación.	

4.3.2.10 MODIFICAR DATOS DE UNA CATEGORÍA DE GASTO

CASO DE USO	MODIFICAR DATOS DE UNA CATEGORÍA DE GASTO
Actores	Usuario
Propósito	Modificar los datos de una Categoría de Gasto de la Base de Datos
Resumen	El usuario accede a la opción de lista de categorías, el usuario selecciona una categoría, y la aplicación le muestra los datos de la Categoría. El usuario modifica los datos de la categoría, y la aplicación guarda los datos modificados en la Base de Datos.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de categorías del menú.	
	2 La aplicación muestra el listado de categorías.
3 El usuario selección una de las categorías del listado de gastos.	
	4 La aplicación muestra el formulario de modificar datos de categoría.
5 El usuario modifica los datos del formulario, y aprieta el botón guardar	
	6 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
7 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 5.	

4.3.2.11 ELIMINAR CATEGORÍA DE GASTO DE LA BASE DE DATOS

CASO DE USO	ELIMINAR CATEGORÍA DE GASTO DE LA BASE DE DATOS
Actores	Usuario
Propósito	Eliminar una categoría de Gasto definitivamente de la base de datos.
Resumen	El usuario accede a la opción de lista de categorías, el usuario selecciona una categoría, y la aplicación le muestra los datos de la categoría. El usuario selecciona la opción de eliminar categoría, y la aplicación elimina la categoría de la Base de Datos.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de categorías del menú.	
	2 La aplicación muestra el listado de categorías.
3 El usuario selección una de las categorías del listado de gastos.	
	4 La aplicación muestra el formulario de modificar datos de categoría.
5 El usuario aprieta el botón eliminar categoría del formulario.	
	6 La aplicación comprueba los datos y elimina la categoría de la base de datos
CURSOS ALTERNATIVOS	
7 La categoría está en uso por un gasto y no se puede borrar. Se le muestra al usuario un mensaje de error, y vuelve a la línea 5.	

4.3.2.12 LISTAR CATEGORÍAS DE GASTO DE LA BASE DE DATOS

CASO DE USO	LISTAR CATEGORÍAS DE GASTO DE LA BASE DE DATOS
Actores	Usuario
Propósito	Listar las categorías de gasto de la base de datos
Resumen	El usuario accede a la opción de lista de categorías, la aplicación le muestra un listado de las categorías de gastos de la base de datos.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de categorías del menú.	
	2 La aplicación muestra el listado de categorías.
CURSOS ALTERNATIVOS	
No hay cursos alternativos.	

4.3.2.13 AÑADIR LUGAR DE GASTO A LA BASE DE DATOS DE FORMA MANUAL

CASO DE USO	AÑADIR LUGAR DE GASTO A LA BASE DE DATOS DE FORMA MANUAL
Actores	Usuario
Propósito	Dar de alta un lugar de gasto en la Base de Datos de forma manual.
Resumen	El usuario accede a la opción de añadir lugar, rellena el formulario con los datos del lugar. La aplicación almacena el lugar en la Base de Datos
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de añadir lugar del menú.	
	2 La aplicación muestra el formulario de añadir lugar de forma manual.

3 El usuario rellena el formulario, y aprieta el botón guardar	
	4 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
5 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 3.	

4.3.2.14 AÑADIR LUGAR DE GASTO A LA BASE DE DATOS POR GEO LOCALIZACIÓN

CASO DE USO	**AÑADIR LUGAR DE GASTO A LA BASE DE DATOS POR GEO LOCALIZACIÓN.**
Actores	Usuario
Propósito	Dar de alta un lugar de gasto en la Base de Datos por geo localización.
Resumen	El usuario accede a la opción de añadir lugar, rellena el formulario con los datos del lugar, y aprieta el botón de geo localización para obtener la situación del lugar. La aplicación almacena el lugar en la Base de Datos
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de añadir lugar del menú.	
	2 La aplicación muestra el formulario de añadir lugar por geo localización.
3 El usuario rellena el formulario, y aprieta el botón de geo localización	
	4 La aplicación obtiene la geo localización del lugar.
5 El usuario aprieta el botón guardar.	
	6 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
5 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 3.	

4.3.2.15 AÑADIR LUGAR DE GASTO A LA BASE DE DATOS POR POSICIÓN EN EL MAPA

CASO DE USO	AÑADIR LUGAR DE GASTO A LA BASE DE DATOS POR POSICIÓN EN EL MAPA.
Actores	Usuario
Propósito	Dar de alta un lugar de gasto en la Base de Datos por posición en el mapa
Resumen	El usuario accede a la opción de añadir lugar por mapa, selecciona un punto, rellena el formulario con los datos del lugar. La aplicación almacena el lugar en la Base de Datos
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de añadir lugar por posición en el mapa del menú.	
	2 La aplicación muestra el formulario de añadir lugar por posición en el mapa.
3 El usuario selección una punto en el mapa, rellena el formulario, y aprieta el botón de guardar.	
	4 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
5 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 3.	

4.3.2.16 MODIFICAR DATOS DE UN LUGAR DE GASTO

CASO DE USO	MODIFICAR DATOS DE UN LUGAR DE GASTO
Actores	Usuario
Propósito	Modificar los datos de un lugar de Gasto de la Base de Datos
Resumen	El usuario accede a la opción de lista de lugares, el usuario selecciona un lugar, y la aplicación le muestra los datos del lugar. El usuario modifica los datos del lugar, y la aplicación guarda los datos modificados en la Base de Datos.
Tipo	Primario y esencial.

CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de lugares del menú.	
	2 La aplicación muestra el listado de lugares.
3 El usuario selección una de los lugares del listado de gastos.	
	4 La aplicación muestra el formulario de modificar datos de lugares.
5 El usuario modifica los datos del formulario, y aprieta el botón guardar	
	6 La aplicación comprueba los datos rellenados y los almacena.
CURSOS ALTERNATIVOS	
7 Alguno de los datos no es correcto, se muestra un error, y se vuelve a la Línea 5.	

4.3.2.17 VISUALIZAR DATOS DE UN LUGAR DE GASTO

CASO DE USO	VISUALIZAR DATOS DE UN LUGAR DE GASTO
Actores	Usuario
Propósito	Mostrar todos los datos de un lugar de Gasto de la Base de Datos
Resumen	El usuario accede a la opción de lista de lugares, el usuario selecciona un lugar, y la aplicación le muestra los datos del lugar.
Tipo	Primario y esencial.

CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de lugares del menú.	
	2 La aplicación muestra el listado de lugares.
3 El usuario selección uno de los lugares del listado de lugares.	

	4 La aplicación muestra el formulario de datos del lugar.
CURSOS ALTERNATIVOS	
5 No existe ningún lugar en la base de datos, al usuario se le muestra un mensaje avisándole de la situación.	

4.3.2.18 ELIMINAR LUGAR DE GASTO DE LA BASE DE DATOS

CASO DE USO	**ELIMINAR LUGAR DE GASTO DE LA BASE DE DATOS**
Actores	Usuario
Propósito	Eliminar un lugar de Gasto definitivamente de la base de datos.
Resumen	El usuario accede a la opción de lista de lugares, el usuario selecciona un lugar, y la aplicación le muestra los datos del lugar. El usuario selecciona la opción de eliminar lugar, y la aplicación elimina el lugar de la Base de Datos.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de lugares del menú.	
	2 La aplicación muestra el listado de lugares.
3 El usuario selección uno de los lugares del listado de gastos.	
	4 La aplicación muestra el formulario de modificar datos de lugar.
5 El usuario aprieta el botón eliminar lugar del formulario.	
	6 La aplicación comprueba los datos y elimina el lugar de la base de datos
CURSOS ALTERNATIVOS	
7 El lugar está en uso por un gasto y no se puede borrar. Se le muestra al usuario un mensaje de error, y vuelve a la línea 5.	

4.3.2.19 LISTAR LUGARES DE GASTO DE LA BASE DE DATOS

CASO DE USO	LISTAR LUGARES DE GASTO DE LA BASE DE DATOS
Actores	Usuario
Propósito	Listar los lugares de gasto de la base de datos
Resumen	El usuario accede a la opción de lista de lugares, la aplicación le muestra un listado de los lugares de gastos de la base de datos.
Tipo	Primario y esencial.
CURSO TÍPICO	
Acción del actor	**Respuesta del sistema**
1 El usuario accede a la opción de lista de lugares del menú.	
	2 La aplicación muestra el listado de lugares.
CURSOS ALTERNATIVOS	
No hay cursos alternativos.	

4.3.3 ESPECIFICACIÓN DE CASOS DE USO

En este apartado especificaremos los casos de uso del sistema más destacados. Para algunos casos se ha añadido el diagrama de secuencia en los casos más interesantes, el resto de casos son muy parecidos y no es necesario repetir los mismos diagramas.

4.3.3.1 AÑADIR GASTO POR TECLADO

Requisitos referenciados: R1.1

Contratos del sistema:

- **Crea gasto.**

Crea_gasto (nombre, valor, iddescripción, idcategoría, lugar, foto)

- **Pre:**
  - El nombre no es nulo.
  - El valor no es nulo.
  - Existe en el sistema una categoría CAT con el identificador idcategoría.
  - Existe en el sistema un lugar LUG con el identificador idlugar.
- **Post:**
  - Se crea en el sistema un nuevo gasto GAS con los valores pasados como referencia.
  - El gasto GAS queda vinculado a categoría CAT, y lugar LUG.

4.3.3.2 AÑADIR GASTO POR RECONOCIMIENTO ÓPTICO DE CARACTERES

Requisitos referenciados: R1.2

- **CONTRATOS DEL SISTEMA:**
  - **Crear gasto**

Crea_gasto (nombre, valor, iddescripción, idcategoría, lugar, foto)

- **Pre:**
  - El nombre no es nulo.
  - El valor no es nulo.
  - Existe en el sistema una categoría CAT con el identificador idcategoría.
  - Existe en el sistema un lugar LUG con el identificador idlugar.
- **Post:**
  - Se crea en el sistema un nuevo gasto GAS con los valores pasados como referencia.
  - El gasto GAS queda vinculado a categoría CAT, y lugar LUG.

4.3.3.3 VISUALIZAR DATOS DE UN GASTO

Requisitos referenciados: R1.5

Contratos del sistema:

- **Consultar gasto**

Consultar_gasto (idgasto)

- **Pre:**
  - Existe en el sistema un gasto GAS con el identificador idgasto.
- **Post:**
  - El sistema obtiene todos los datos del gasto GAS y los muestra en pantalla.

4.3.3.4 MODIFICAR DATOS DE UN GASTO

Requisitos referenciados: R1.3

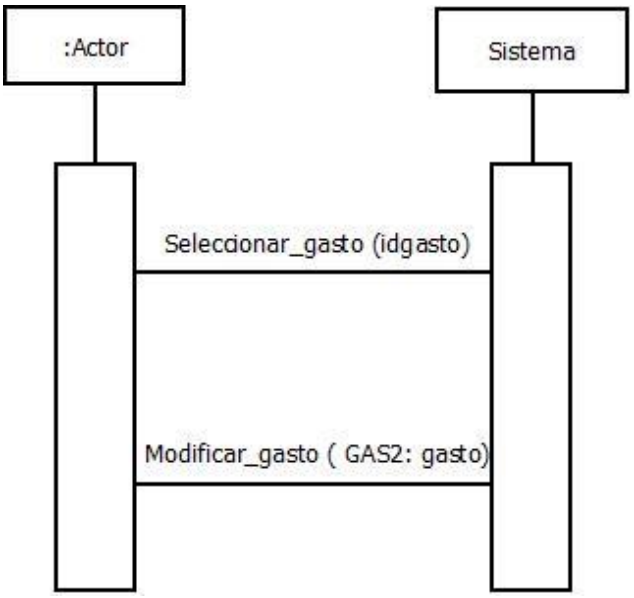
- **CONTRATOS DEL SISTEMA:**
  - Seleccionar gasto
  - Modificar gasto

Seleccionar_gasto (idgasto)

- **Pre:**
  - Existe en el sistema un gasto GAS con el identificador idgasto.
- **Post:**
  - El sistema obtiene todos los datos del gasto GAS y los muestra en pantalla.
  - El gasto GAS queda activado.

Modificar_gasto(GAS2: gasto)

- **Pre:**
  - Existe en el sistema un Gasto GAS activado.
- **Post:**
  - El gasto activado se ha modificado con los datos del gasto GAS2.
  - No existe en el sistema un Gasto GAS activado.



4.3.3.5 ELIMINAR GASTO DE LA BASE DE DATOS

Requisitos referenciados: R1.4

- **CONTRATOS DEL SISTEMA:**
  - **Eliminar gasto**

Eliminar_gasto (idgasto)

- **Pre:**
  - Existe en el sistema un gasto GAS con el identificador idgasto.
- **Post:**
  - El gasto GAS se ha eliminado.

4.3.3.6 Listar gastos de la base de datos.

Requisitos referenciados: R1.6

- **CONTRATOS DEL SISTEMA:**
  - **Generar listado gasto**

Generar_listado_gasto ()

- **Pre:**
  - Existe en el sistema gastos.
- **Post:**
  - El sistema obtiene todos los datos de los gastos y los muestra en pantalla.

4.3.3.7 MOSTRAR RESUMEN DE GASTOS DE LA BASE DE DATOS

Requisitos referenciados: R1.6

- **CONTRATOS DEL SISTEMA:**
  - **Generar resumen gasto**

Generar_resumen_gasto (numero)

- **Pre:**
  - Existe en el sistema gastos.
  - Número no es negativo o 0.
- **Post:**
  - El sistema obtiene una cantidad de gastos determinada por el número y los muestra en pantalla.

4.3.3.8 AÑADIR CATEGORÍA DE GASTO A LA BASE DE DATOS.

Requisitos referenciados: R2.1

- **CONTRATOS DEL SISTEMA:**
  - **Crea categoría**

Crea_categoria (nombre, descripción, color)

- **Pre:**
  - El nombre no es nulo.
  - El color no es nulo.
- **Post:**
  - Se crea en el sistema una nueva categoría CAT.

4.3.3.9 VISUALIZAR DATOS DE UNA CATEGORÍA DE GASTO

Requisitos referenciados: R2.4

- **CONTRATOS DEL SISTEMA:**
  - **Consultar categoría**

Consultar_categoria (idcategoria)

- **Pre:**
  - Existe en el sistema una categoría CAT con el identificador idcategoria.
- **Post:**
  - El sistema obtiene todos los datos de la categoría CAT y los muestra en pantalla.

4.3.3.10 MODIFICAR DATOS DE UNA CATEGORÍA DE GASTO

Requisitos referenciados: R2.2

- **CONTRATOS DEL SISTEMA:**
  - **Seleccionar categoría**
  - **Modificar categoría**

Seleccionar_categoria (idcategoria)

- **Pre:**
  - Existe en el sistema una categoría CAT con el identificador idcategoria.
- **Post:**
  - El sistema obtiene todos los datos de la categoría CAT y los muestra en pantalla.

- La categoría CAT queda activada.

Modificar_categoria (CAT2: categoría)

- **Pre:**
  - Existe en el sistema una categoría CAT activada.
  - Los datos de la categoría CAT2 son correctos/validos.
- **Post:**
  - La categoría activada se ha modificado con los datos de la categoría CAT2.
  - No existe en el sistema una categoría CAT activada.

4.3.3.11 ELIMINAR CATEGORÍA DE GASTO DE LA BASE DE DATOS

Requisitos referenciados: R2.3

- **CONTRATOS DEL SISTEMA:**
  - **Eliminar categoría**

Eliminar_categoria (idcategoria)

- **Pre:**
  - Existe en el sistema una categoría CAT con el identificador idcategoria.
  - No existe ningún gasto vinculado a la categoría con identificador idcategoria.
- **Post:**
  - La categoría CAT se ha eliminado.

4.3.3.12 LISTAR CATEGORÍAS DE GASTO DE LA BASE DE DATOS

Requisitos referenciados: R2.5

- **CONTRATOS DEL SISTEMA:**
  - **Generar listado categoría.**

Generar_listado_categoria ()

- **Pre:**
  - Existe en el sistema categorías.
- **Post:**
  - El sistema obtiene todos los datos de las categorías y los muestra en pantalla.

4.3.3.13 AÑADIR LUGAR DE GASTO A LA BASE DE DATOS DE FORMA MANUAL

Requisitos referenciados: R3.1

- **CONTRATOS DEL SISTEMA:**
  - **Crea lugar**

Crea_lugar (nombre, descripción, latitud, longitud)

- **Pre:**
  - El nombre no es nulo.
- **Post:**
  - Se crea en el sistema un nuevo lugar LUG.

4.3.3.14 AÑADIR LUGAR DE GASTO A LA BASE DE DATOS POR GEO LOCALIZACIÓN

Requisitos referenciados: R3.2

- **CONTRATOS DEL SISTEMA:**
  - **Crea lugar**

Crea_lugar (nombre, descripción, latitud, longitud)

- **Pre:**
  - El nombre no es nulo.
- **Post:**
  - Se crea en el sistema un nuevo lugar LUG.

4.3.3.15 AÑADIR LUGAR DE GASTO A LA BASE DE DATOS POR POSICIÓN EN EL MAPA

Requisitos referenciados: R3.3

- **CONTRATOS DEL SISTEMA:**
  - **Crea lugar**

Crea_lugar (nombre, descripción, latitud, longitud)

- **Pre:**
  - El nombre no es nulo.
- **Post:**
  - Se crea en el sistema un nuevo lugar LUG.

4.3.3.16 MODIFICAR DATOS DE UN LUGAR DE GASTO

Requisitos referenciados: R3.3

- **CONTRATOS DEL SISTEMA:**
  - **Seleccionar lugar**
  - **Modificar lugar**

Seleccionar_lugar (idlugar)

- **Pre:**
  - Existe en el sistema un lugar LUG con el identificador idlugar.
- **Post:**
  - El sistema obtiene todos los datos del lugar LUG y los muestra en pantalla.
  - El lugar LUG queda activado.

Modificar_lugar (LUG2: lugar)

- **Pre:**
  - Existe en el sistema un lugar LUG activado.
- **Post:**
  - El lugar activado se ha modificado con los datos del lugar LUG2.
  - No existe en el sistema un lugar LUG activado.

4.3.3.17 VISUALIZAR DATOS DE UN LUGAR DE GASTO

Requisitos referenciados: R3.5

- **CONTRATOS DEL SISTEMA:**
  - **Consultar lugar**

Consultar_lugar (idlugar)

- **Pre:**
  - Existe en el sistema un lugar LUG con el identificador idlugar.
- **Post:**
  - El sistema obtiene todos los datos del lugar LUG y los muestra en pantalla.

4.3.3.18 ELIMINAR LUGAR DE GASTO DE LA BASE DE DATOS

Requisitos referenciados: R3.4

- **CONTRATOS DEL SISTEMA:**
  - **Eliminar lugar**

Eliminar_lugar (idlugar)

- **Pre:**
  - Existe en el sistema un lugar LUG con el identificador idlugar.
  - No existe ningún gasto vinculado al lugar con identificador idlugar.
- **Post:**
  - El lugar LUG se ha eliminado.

4.3.3.19 LISTAR LUGARES DE GASTO DE LA BASE DE DATOS

Requisitos referenciados: R3.6

- **CONTRATOS DEL SISTEMA:**
  - **Generar listado lugar.**

Generar_listado_lugar ()

- **Pre:**
  - Existe en el sistema lugares.
- **Post:**
  - El sistema obtiene todos los datos de los lugares y los muestra en pantalla.

4.3.3.20 MÉTODOS PARA COMPROBAR LAS PREs

Para cumplir las condiciones del sistema se necesitaran los siguientes métodos para comprobar que las **PRE:**

Existe_Gasto(idgasto): Boolean

Descripción: Este método devuelve cierto si existe un gasto GAS:Gasto en Lugar, tal que idgasto de GAS sea igual a idgasto.

Existe_Categoria(idcategoria): Boolean

Descripción: Este método devuelve cierto si existe una categoría CAT:Categoria en Categoría, tal que idcategoria de CAT sea igual a idcategoria.

Existe_Lugar(idlugar): Boolean

Descripción: Este método devuelve cierto si existe un lugar LUG:Lugar en Lugar, tal que idlugar de LUG sea igual a idlugar.

Comprobar_nombregasto(nombre): Boolean

Descripción: Este método devuelve cierto si el nombre:String no es nulo y son caracteres válidos.

Comprobar_valorgasto(valor): Boolean

Descripción: Este método devuelve cierto si el valor:Double no es nulo, es diferente a 0 y no es negativo.

Existen_gastos(): Boolean

Descripción: Este método devuelve cierto si en Gastos existe algún gasto.

Comprobar_nombrecategoria(nombre): Boolean

Descripción: Este método devuelve cierto si el nombre:String no es nulo y son caracteres válidos.

Comprobar_colorcategoria(color): Boolean

Descripción: Este método devuelve cierto si el color:Int no es nulo.

Categoria_enuso(idcategoria): Boolean

Descripción: Este método devuelve cierto si existe un GAS:Gasto en Gastos, tal que idcategoria de GAS sea igual a idcategoria.

Comprobar_nombrelugar(nombre): Boolean

Descripción: Este método devuelve cierto si el nombre:String no es nulo y son caracteres válidos.

Lugar_enuso(idlugar): Boolean

Descripción: Este método devuelve cierto si existe un GAS:Gasto en Gastos, tal que idlugar de GAS sea igual a idlugar.

4.4 DIAGRAMAS DE COLABORACIÓN

En este apartado determinaremos un conjunto de componentes que colaboraran entre sí. Su comportamiento estará basado en lo que pide las especificaciones.

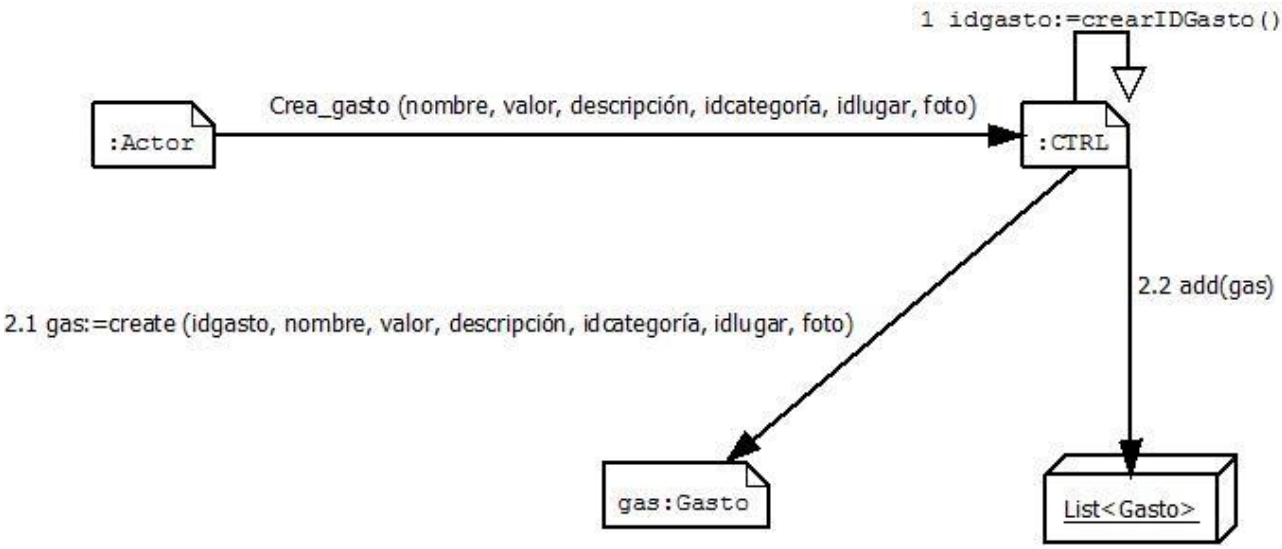
Estos diagramas se realizan para facilitar la implementación, ya que después de ellos solo tendremos que resolver las distintas peculiaridades del lenguaje de programación que hemos optado a utilizar, en este caso Java.

Los diagramas de colaboración de casos de uso de listado y resumen los he omitido debido que son repeticiones de los casos de uso de consulta

4.4.1 DIAGRAMAS

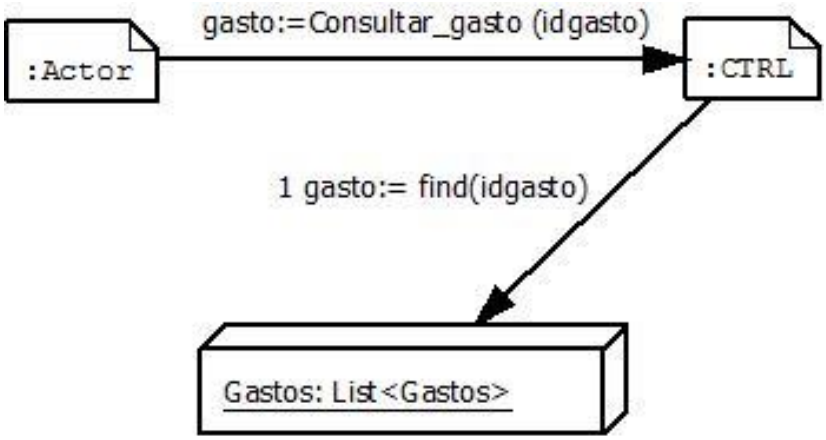
4.4.1.1 AÑADIR GASTO

- **Crea gasto**



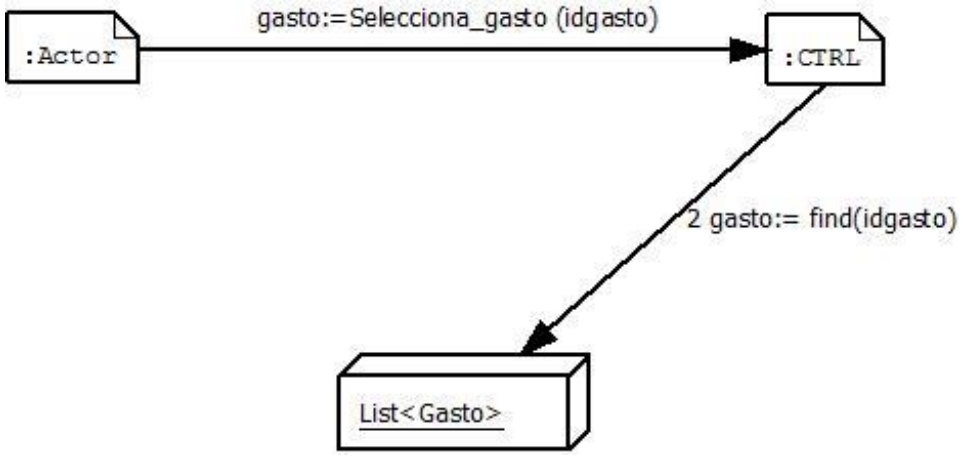
4.4.1.2 VISUALIZAR DATOS DE UN GASTO

- **Consultar_gasto (idgasto)**

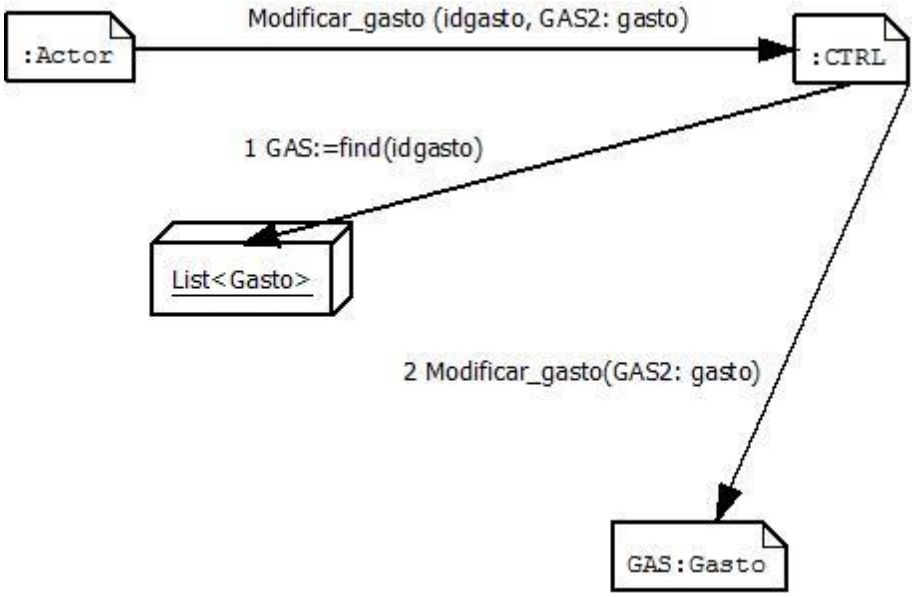


4.4.1.3 MODIFICAR DATOS DE UN GASTO

- **Seleccionar_gasto (idgasto)**

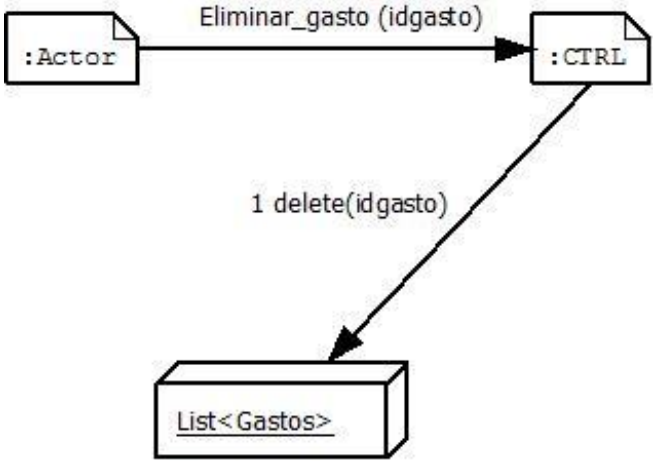


- **Modificar_gasto (GAS2: gasto)**



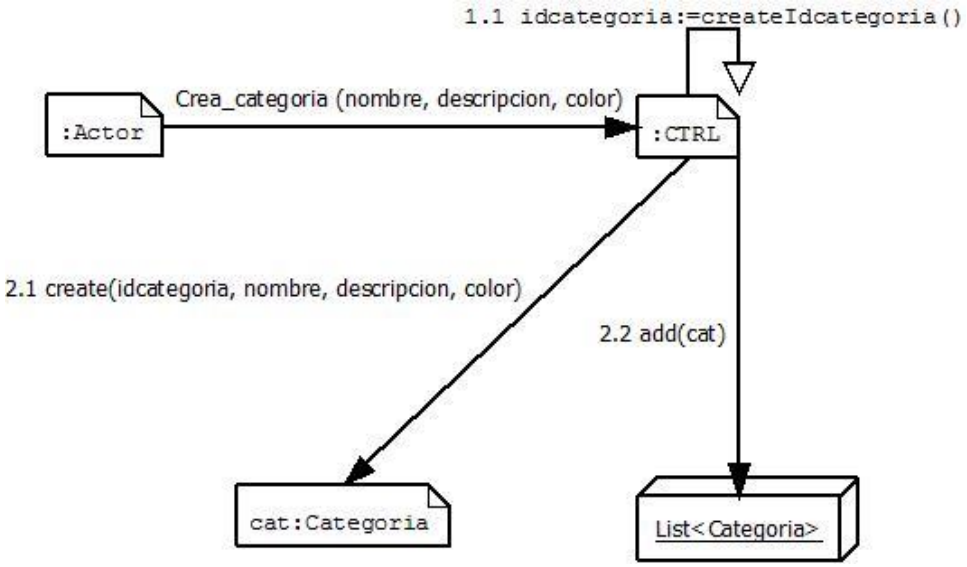
4.4.1.4 ELIMINAR GASTO DE LA BASE DE DATOS

- **Eliminar_gasto (idgasto)**



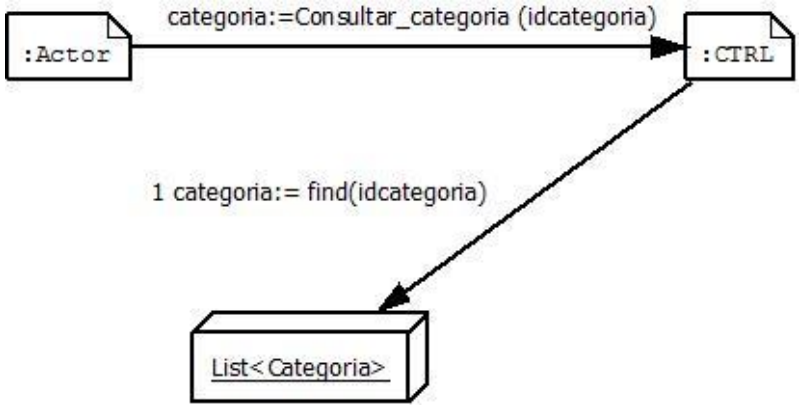
4.4.1.5 AÑADIR CATEGORÍA DE GASTO A LA BASE DE DATOS

- **Crea_categoria (nombre, descripción, color)**



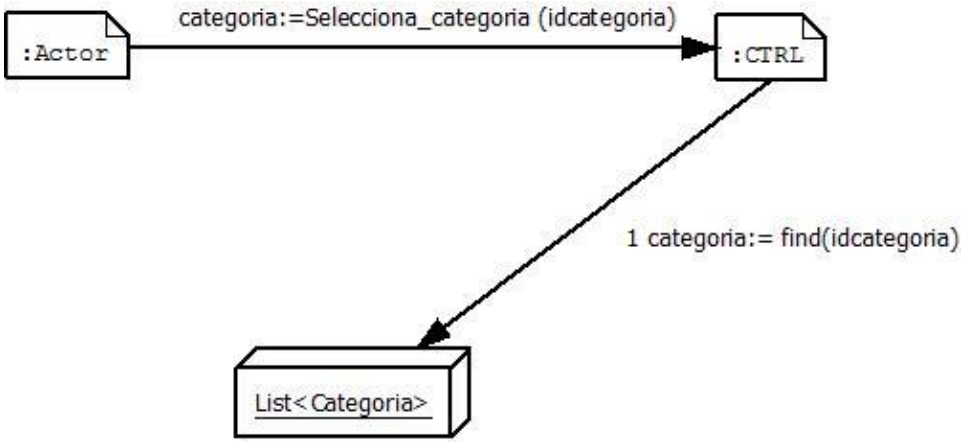
4.4.1.6 VISUALIZAR DATOS DE UNA CATEGORÍA DE GASTO

- **Consultar_categoria (idcategoria)**

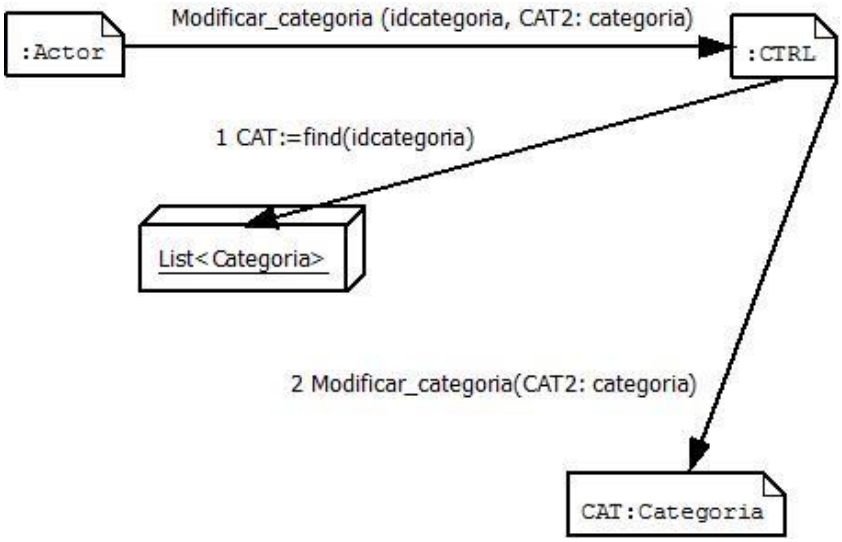


4.4.1.7 MODIFICAR DATOS DE UNA CATEGORÍA DE GASTO

- **Seleccionar_categoria (idcategoria)**

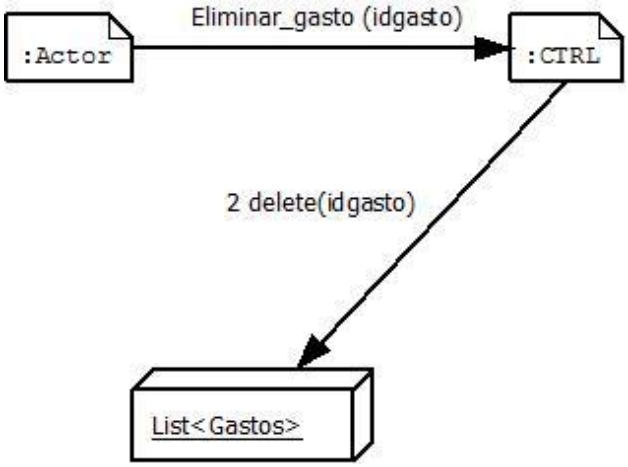


- **Modificar_categoria (CAT2: categoría)**



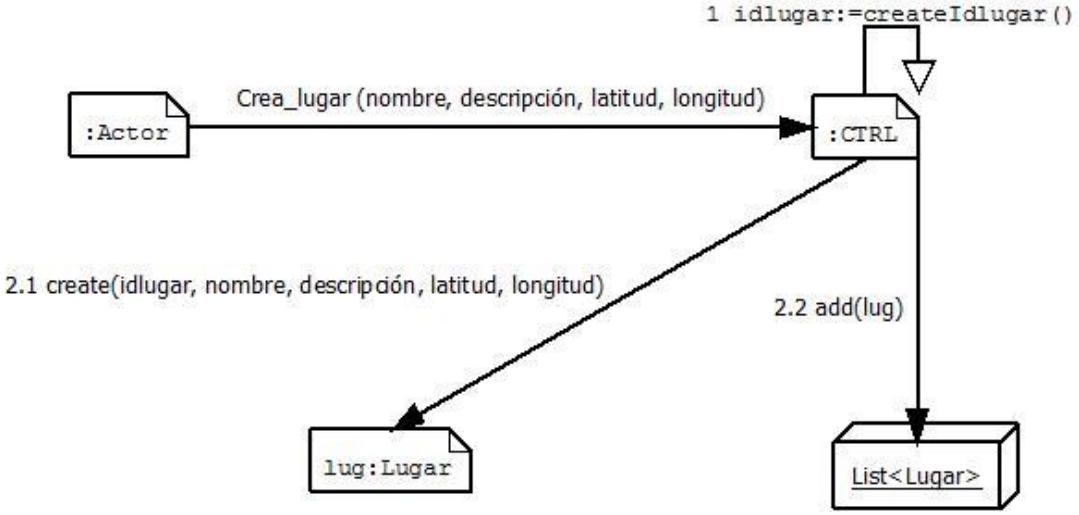
4.4.1.8 ELIMINAR CATEGORÍA DE GASTO DE LA BASE DE DATOS

- **Eliminar_categoria (idcategoria)**



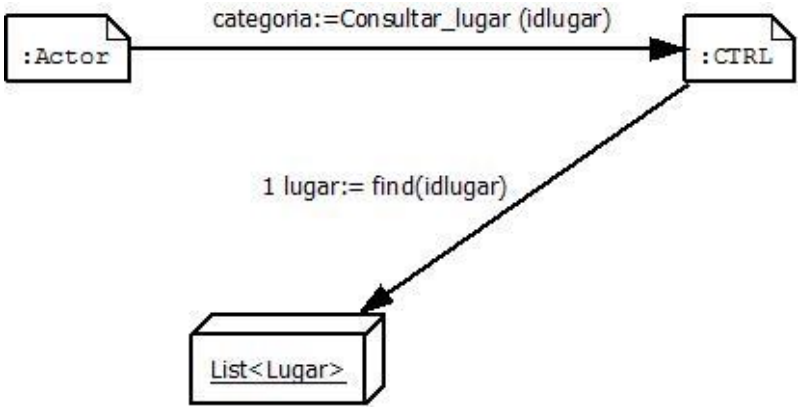
4.4.1.9 AÑADIR LUGAR DE GASTO A LA BASE DE DATOS

- **Crea_lugar (nombre, descripción, latitud, longitud)**



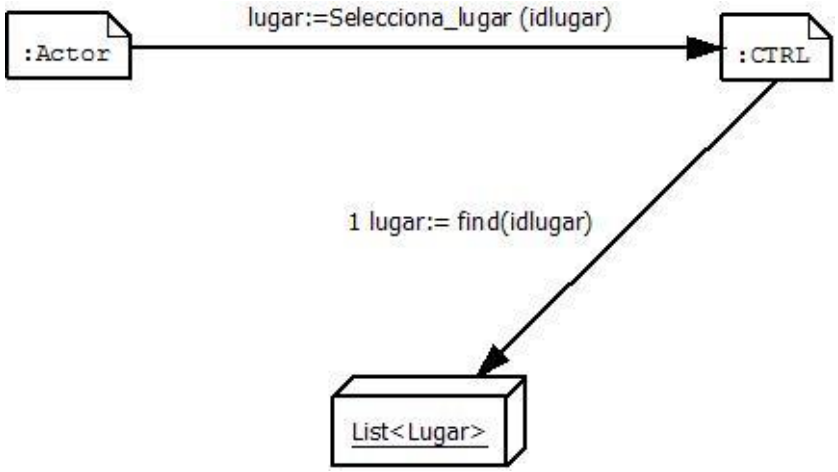
4.4.1.10 VISUALIZAR DATOS DE UN LUGAR DE GASTO

- **Consultar_lugar (idlugar)**

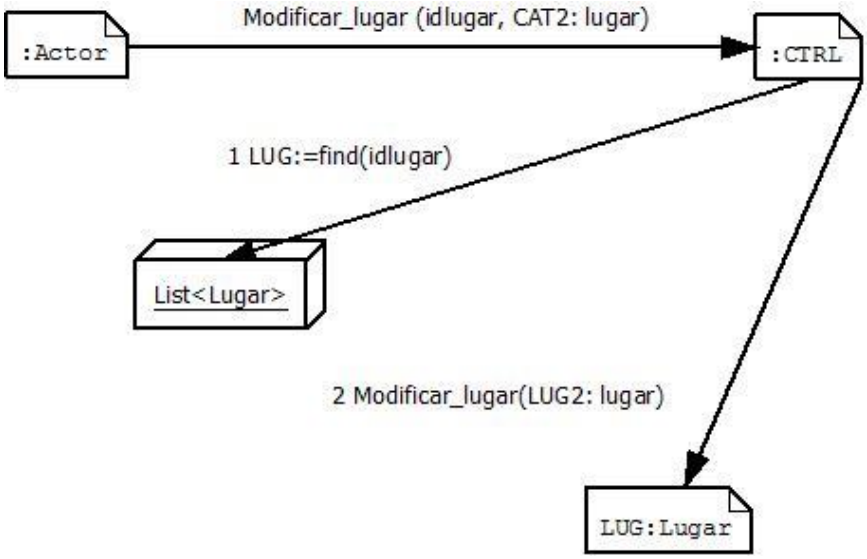


4.4.1.11 MODIFICAR DATOS DE UN LUGAR DE GASTO

- **Seleccionar_lugar (idlugar)**

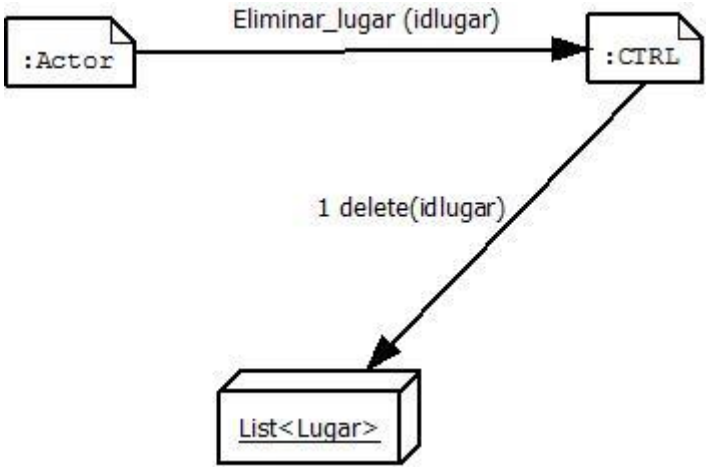


- **Modificar_lugar (LUG2: lugar)**

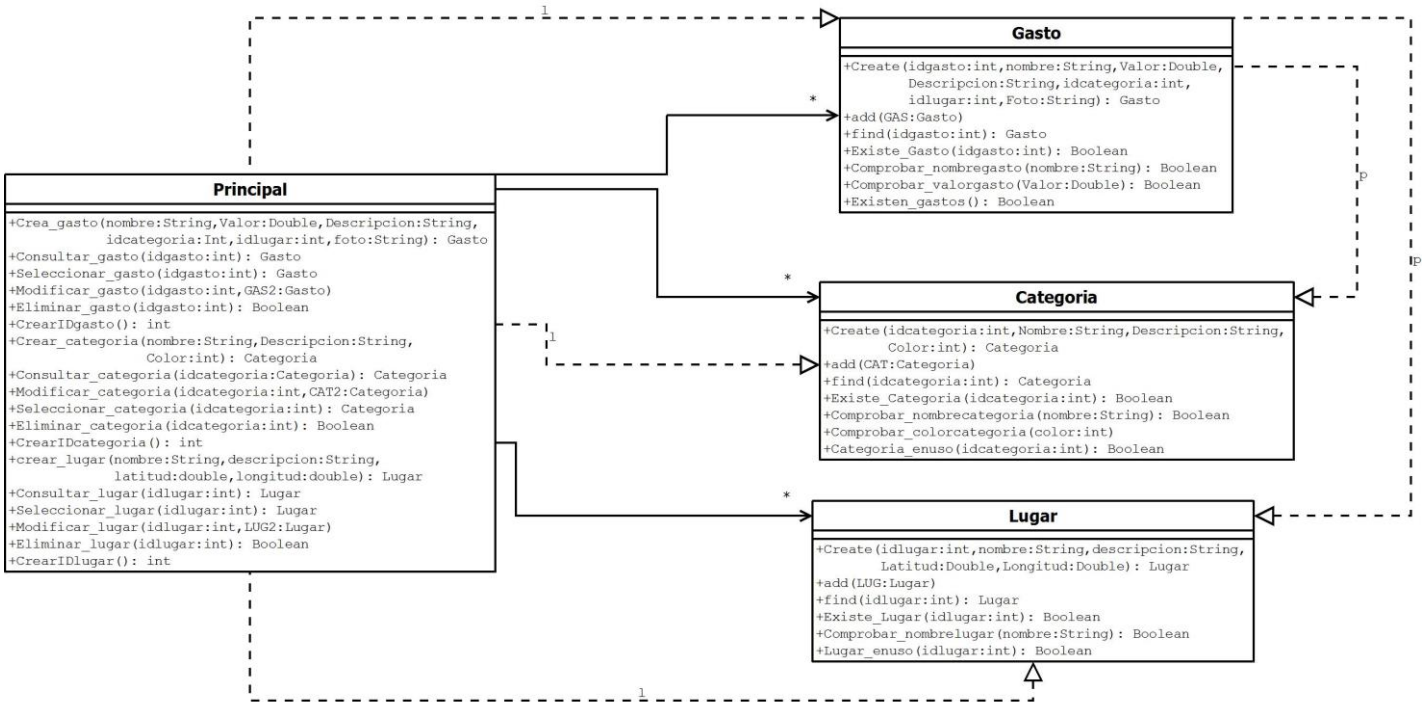


4.4.1.12 ELIMINAR LUGAR DE GASTO DE LA BASE DE DATOS

- **Eliminar_lugar (idlugar)**



4.4.2 MODELO DE COMPONENTES



- **LISTADO DE COMPONENTES**

CATEGORÍA	
ATRIBUTOS	• Int: idcategoria • String: Nombre • String: Descripción • Int: Color • List<Categoría>: Categoría
MÉTODOS	• Categoría: find(int: idcategoria) • Boolean: add(Categoría CAT) • Categoria: Create(int: idcategoria, String: Nombre, String: Descripción, Int: Color) • Boolean: Existe_Categoria(Int: idcategoria) • Boolean: Comprobar_nombrecategoria(String: Nombre) • Comprobar_colorcategoria(Int: Color) • Boolean: Categoria_enuso(Int: idcategoria)
VISIBILIDADES	• Local: Principal • Parámetro: Gasto

LUGAR	
ATRIBUTOS	• Int: idlugar • String: Nombre • String: Descripción • Long: Latitud • Long: Longitud • List<Lugar>: Lugar
MÉTODOS	• Lugar: find(int: idlugar) • Boolean: add(Lugar LUG) • Lugar: Create(Int: idlugar, String: Nombre, String: Descripción, Long: Latitud, Long: Longitud) • Boolean: Existe_Lugar(Int: idlugar) • Boolean: Comprobar_nombrelugar(String: Nombre) • Boolean: Lugar_enuso(Int: idlugar)
VISIBILIDADES	• Local: Principal • Parámetro: Gasto

GASTO	
ATRIBUTOS	• Int: idgasto • String: Nombre • String: Descripción • Double: Valor • Int: Idcategoria • Int: Idlugar • Long: Fecha • String: Foto • List<Gasto>: Gasto
MÉTODOS	• Gasto: find(int: idgasto) • Boolean: add(Gasto GAS) • Gasto: Create(Int: idgasto, String: Nombre, String: Descripción, Double: Valor, Int: Idcategoria, Int: Idlugar, Long: Fecha, String: Foto) • Boolean: Comprobar_nombregasto(String: Nombre) • Boolean: Comprobar_valorgasto(Double: Valor) • Boolean: Existen_gastos()
VISIBILIDADES	• Local: Principal • Parámetro: Categoría • Parámetro: Lugar

PRINCIPAL (CONTROLADOR DEL SISTEMA)	
ATRIBUTOS	
MÉTODOS	• Boolean: Crea_categoria(String nombre, String descripción, Int color) • Boolean: Modificar_categoria(int idcategoria, String nombre, String descripción, Int color) • Boolean: eliminar_categoria(int idcategoria) • Categoría: consultar_categoria(int idcategoria) • Boolean: comprobar_categoria(int idcategoria) • Boolean: Crea_lugar(String nombre, String descripción, Long latitud, Long longitud) • Boolean: Modificar_lugar(int idlugar, String nombre, String descripción, Int color) • Boolean: eliminar_lugar(int idlugar) • Lugar: consultar_lugar(int idlugar) • Boolean: comprobar_lugar(int idlugar) • Boolean: Crea_Gasto(String nombre, String descripción, Double valor, int idcategoria, int idlugar, Long fecha, String Foto) • Boolean: Modificar_Gasto(int idgasto, String descripción, Double valor, int idcategoria, int idlugar, Long fecha, String Foto) • Boolean: eliminar_Gasto(int idgasto)
VISIBILIDADES	• Local: Categoría • Local: Lugar • Local: Gasto • Atributo: Categoría • Atributo: Lugar • Atributo: Gasto

4.5 MODELO RELACIONAL

En este apartado de la memoria, definiremos las diferentes tablas de nuestra base de datos, correspondiente al modelo conceptual que hemos realizado antes.

Recordamos que Android utiliza SQLite como gestor de Base de Datos.

- **TABLA GASTOS**

En esta tabla almacenaremos los detalles de los gastos del usuario, los campos de la tabla son los siguientes:

IdGasto	INTEGER, PRIMARY KEY, NOT NULL, UNIQUE, AUTOINCREMENT
Nombre	TEXT, NOT NULL
Valor	REAL, NOT NULL
Descripción	TEXT
Categoría	INTEGER, NOT NULL, FOREIGN KEY de la tabla CATEGORIAS
Lugar	INTEGER, NOT NULL, FOREIGN KEY de la tabla LUGARES
Fecha	INTEGER, NOT NULL
Foto	TEXT

- **Nombre:** Es el nombre que tendrá el gasto. Es un campo tipo TEXT, y por obligación tiene que tener un valor y no puede ser nulo.
- **Valor:** Es el valor monetario del gasto. Tiene que ser tipo REAL, ya que necesitamos poder guardar valores decimales.
- **Descripción**: Es la descripción del gasto. Es un campo tipo TEXT, y es un campo opcional.
- **Categoría**: Es la categoría a la cual pertenece el gasto. Es un campo tipo INTEGER, por obligación tiene que tener un valor y no puede ser nulo, hace referencia a la tabla CATEGORIAS.
- **Lugar**: Es el lugar donde se ha realizado el gasto. Es un campo tipo INTEGER, por obligación tiene que tener un valor y no puede ser nulo, hace referencia a la tabla LUGARES.
- **Fecha:** Es la fecha y hora de cuando se ha realizado el gasto. Es un campo tipo INTEGER, y por obligación tiene que tener un valor y no puede ser nulo. Debido a algunas limitaciones de SQLite, y para simplificar la implementación, he decido guardar la fecha en milisegundos, Java permite de forma fácil obtener una fecha en milisegundos.
- **Foto**: Es la dirección dentro de la memoria del teléfono donde está la foto asociada al gasto. Es un campo tipo TEXT, y es un campo opcional.

- **TABLA CATEGORIAS**

En esta tabla almacenaremos los detalles de los gastos del usuario, los campos de la tabla son los siguientes:

IdCategoria	INTEGER, PRIMARY KEY, NOT NULL, UNIQUE, AUTOINCREMENT
Nombre	TEXT, NOT NULL
Descripción	TEXT
Color	INTEGER, NOT NULL

- **Nombre:** Es el nombre que tendrá el gasto. Es un campo tipo TEXT, y por obligación tiene que tener un valor y no puede ser nulo.
- **Descripción**: Es la descripción del gasto. Es un campo tipo TEXT, y es un campo opcional.
- **Color**: Es el color que tendrá la categoría, para poder clasificarla de una forma más visual. Es un campo tipo INTEGER, y por obligación tiene que tener un valor y no puede ser nulo.

- **TABLA LUGARES**

En esta tabla almacenaremos los detalles de los gastos del usuario, los campos de la tabla son los siguientes:

IdLugar	INTEGER, PRIMARY KEY, NOT NULL, UNIQUE, AUTOINCREMENT
Nombre	TEXT, NOT NULL
Descripción	TEXT
Latitud	REAL, NOT NULL
Longitud	REAL, NOT NULL

- **Nombre:** Es el nombre que tendrá el gasto. Es un campo tipo TEXT, y por obligación tiene que tener un valor y no puede ser nulo.
- **Descripción:** Es la descripción del gasto. Es un campo tipo TEXT, y es un campo opcional.
- **Latitud:** Es la latitud de la coordenada donde está situado el lugar. Es un campo tipo REAL, y por obligación tiene que tener un valor y no puede ser nulo. Las APIs de Google Maps devuelven 2 valores reales al usuario con las coordenadas del lugar.
- **Longitud:** Es la longitud de la coordenada donde está situado el lugar. Es un campo tipo REAL, y por obligación tiene que tener un valor y no puede ser nulo. Las APIs de Google Maps devuelven 2 valores reales al usuario con las coordenadas del lugar.

- **TABLA PREFERENCIAS**

IdPreferencias	INTEGER, PRIMARY KEY, NOT NULL, UNIQUE, AUTOINCREMENT
NumGastos	INTEGER, NOT NULL
Presupuesto	INTEGER, NOT NULL

- **NumGastos:** Es el número de gastos que se mostraran en el resumen de. Es un campo tipo INTEGER, y por obligación tiene que tener un valor y no puede ser nulo.
- **Presupuesto:** Es el valor máximo de presupuesto de gastos del usuario. Es un campo tipo INTEGER, y por obligación tiene que tener un valor y no puede ser nulo.

- **TABLA IDIOMAS**

idIdioma	INTEGER, PRIMARY KEY, NOT NULL, UNIQUE, AUTOINCREMENT
Nombre	TEXT, NOT NULL
Dirección	TEXT, NOT NULL
NombreFicheroComprimido	TEXT, NOT NULL
NombreFicheroFinal	TEXT, NOT NULL
Instalado	INTEGER, NOT NULL

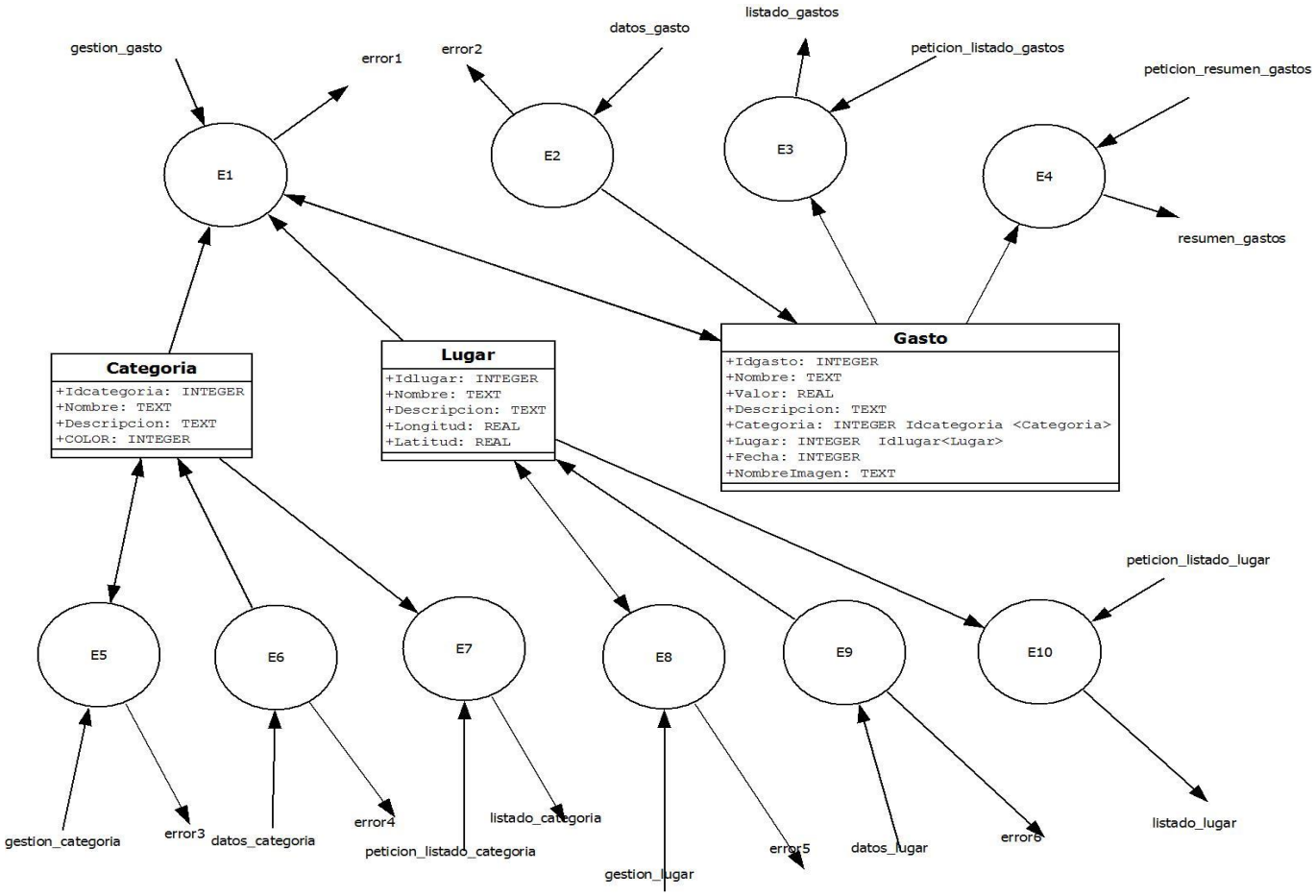
- **Nombre**: Es el nombre que tendrá el Idioma. Es un campo tipo TEXT, y por obligación tiene que tener un valor y no puede ser nulo.
- **Dirección**: Es la dirección de internet o URL del idioma. Es un campo tipo TEXT, y por obligación tiene que tener un valor y no puede ser nulo.
- **NombreFicheroComprimido:** Es el nombre del fichero comprimido al que apunta la Dirección. Es un campo tipo TEXT, y por obligación tiene que tener un valor y no puede ser nulo.
- **NombreFicheroFinal:** Es el nombre del fichero descomprimido. Es un campo tipo TEXT, y por obligación tiene que tener un valor y no puede ser nulo.
- **Instalado:** Instalado nos indica si un idioma está instalado en el sistema. Es un campo tipo INTEGER, y por obligación tiene que tener un valor y no puede ser nulo.

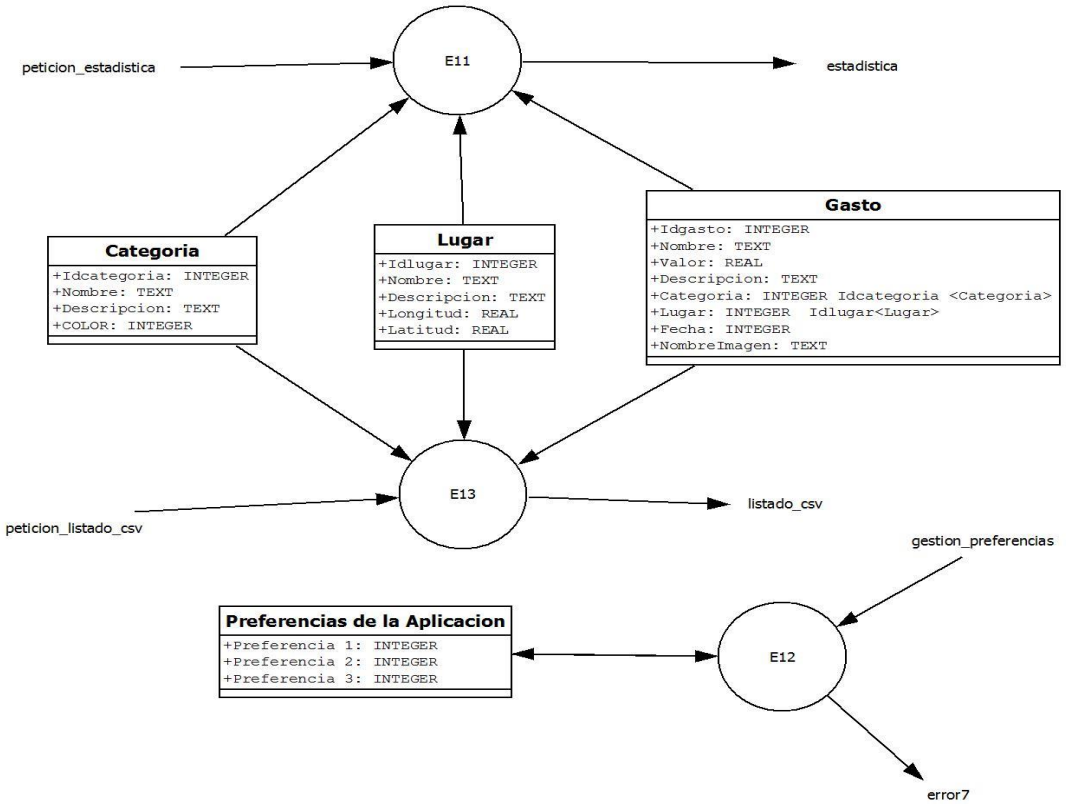
4.6 MODELO DE PROCESOS

4.6.1 LISTADO DE EVENTOS

- E1 Gestión de un gasto (baja, modificación)
- E2 Alta de un gasto
- E3 Generar listado de gastos
- E4 Generar resumen de gastos
- E5 Gestión de una categoría (baja, modificación)
- E6 Alta de una categoría
- E7 Generar listado de categorías
- E8 Gestión de un lugar (baja, modificación)
- E9 Alta de un lugar
- E10 Generar listado de lugares
- E11 Generar Estadística
- E12 Gestión Preferencias (modificación)
- E13 Generar datos exportación a CSV.

4.6.2 DFD NIVEL 1





4.6.3 DEFINICIÓN DE FLUJOS

E1 gestion_gasto = IdGasto + op + categoría + lugar + (nombre + descripción + valor + fecha + foto)

Error1= (¬∃ IdGasto | ¬∃ categoría | ¬∃ lugar | ¬∃ op | ¬ ok | ok)

E2 datos_gasto = nombre + valor + descripción + categoría + lugar + foto

Error2 = (¬∃ IdGasto | ¬∃ categoría | ¬∃ lugar | ¬∃ nombre | ¬∃ valor | ¬∃ fecha | ¬∃ op | ¬ ok | ok)

E3 listados_gastos = 1 { nombre + valor + descripción + categoría + lugar + foto } N

E4 resumen_gastos = 1 { nombre + valor + descripción + categoría + lugar + foto }N

E5 gestion_categoria = idcategoria + op + (nombre + descripción + color)

Error3 = (¬∃ idcategoria | ¬∃ nombre | ¬∃ color | ¬∃ op | ¬ ok | ok)

E6 datos_categoria = nombre + descripción + color

Error4 = (¬∃ nombre | ¬∃ color | ¬ ok | ok)

E7 listado_categoria = 1 { nombre + descripción + color } N

E8 gestion_lugar = idlugar + op + (nombre + descripción + latitud + longitud)

Error5 = (¬∃ idlugar | ¬∃ nombre | ¬∃ latitud | ¬∃ longitud | ¬∃ op | ¬ ok | ok)

E9 datos_lugar = nombre + descripción + latitud + longitud

Error6 = (¬∃ nombre | ¬∃ latitud | ¬∃ longitud ¬ ok | ok)

E10 listado_categoria = 1 { nombre + descripción + latitud + longitud } N

E11 estadística = 1 { nombre + valor + descripción + categoría + lugar + foto } N

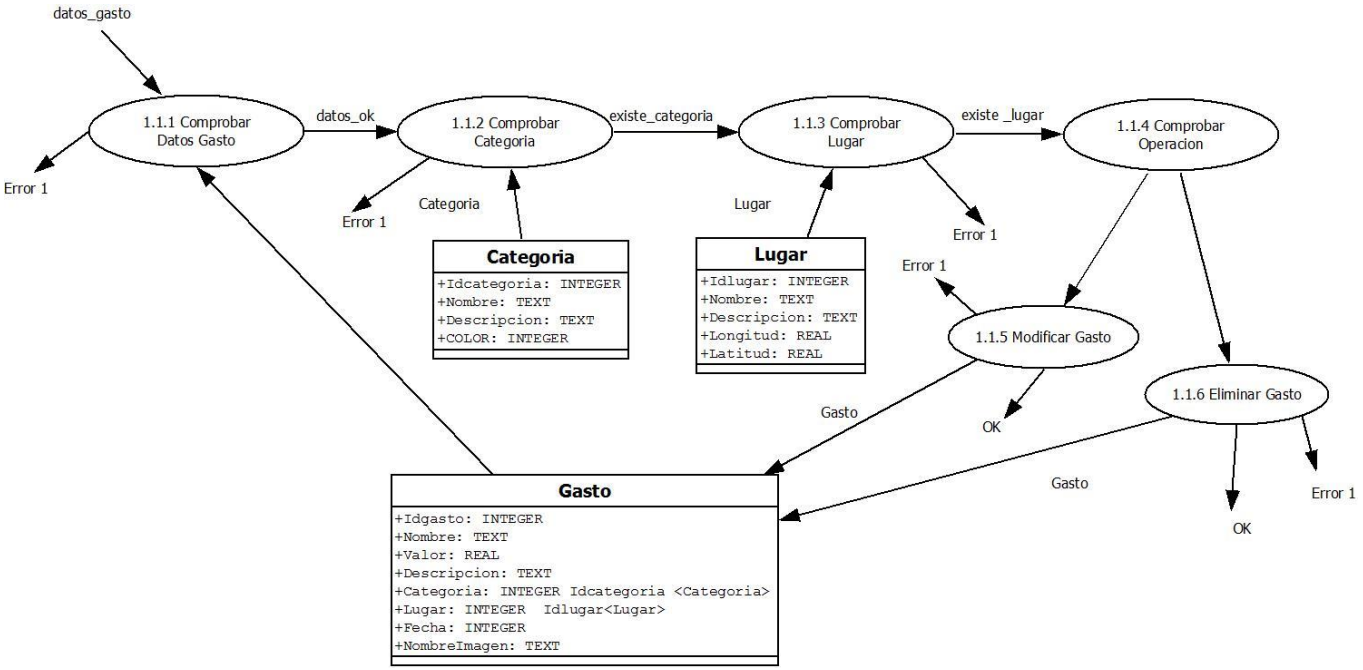
E12 listado_csv = 1 { nombre + valor + descripción + categoría + lugar + foto } N

E13 gestion_preferencias = IdPreferencias + op + (preferencia1 + preferencia2 + preferencia3)

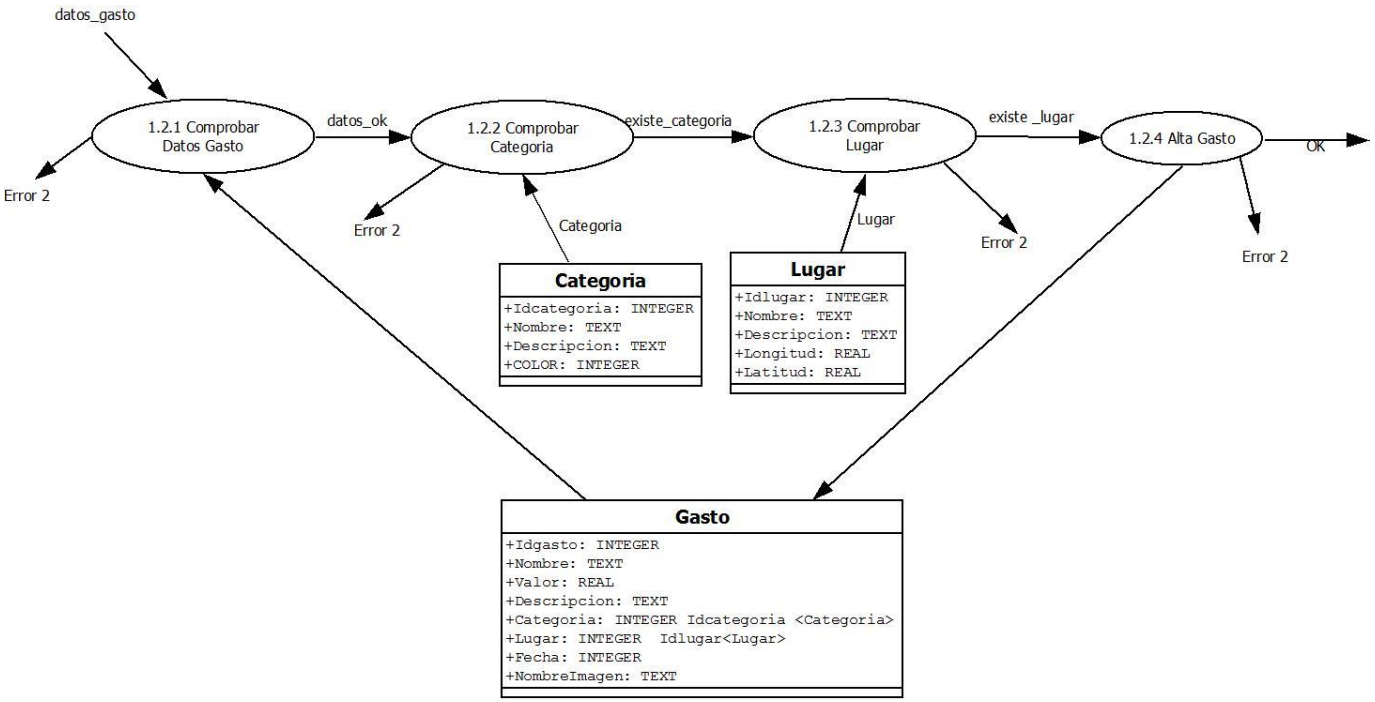
Error 7 = (¬∃ IdPreferencias | ¬∃ preferencia1 | ¬∃ preferencia2 | ¬∃ preferencia3 | ¬∃ op | ¬ ok | ok)

4.6.4 DFD NIVEL 2

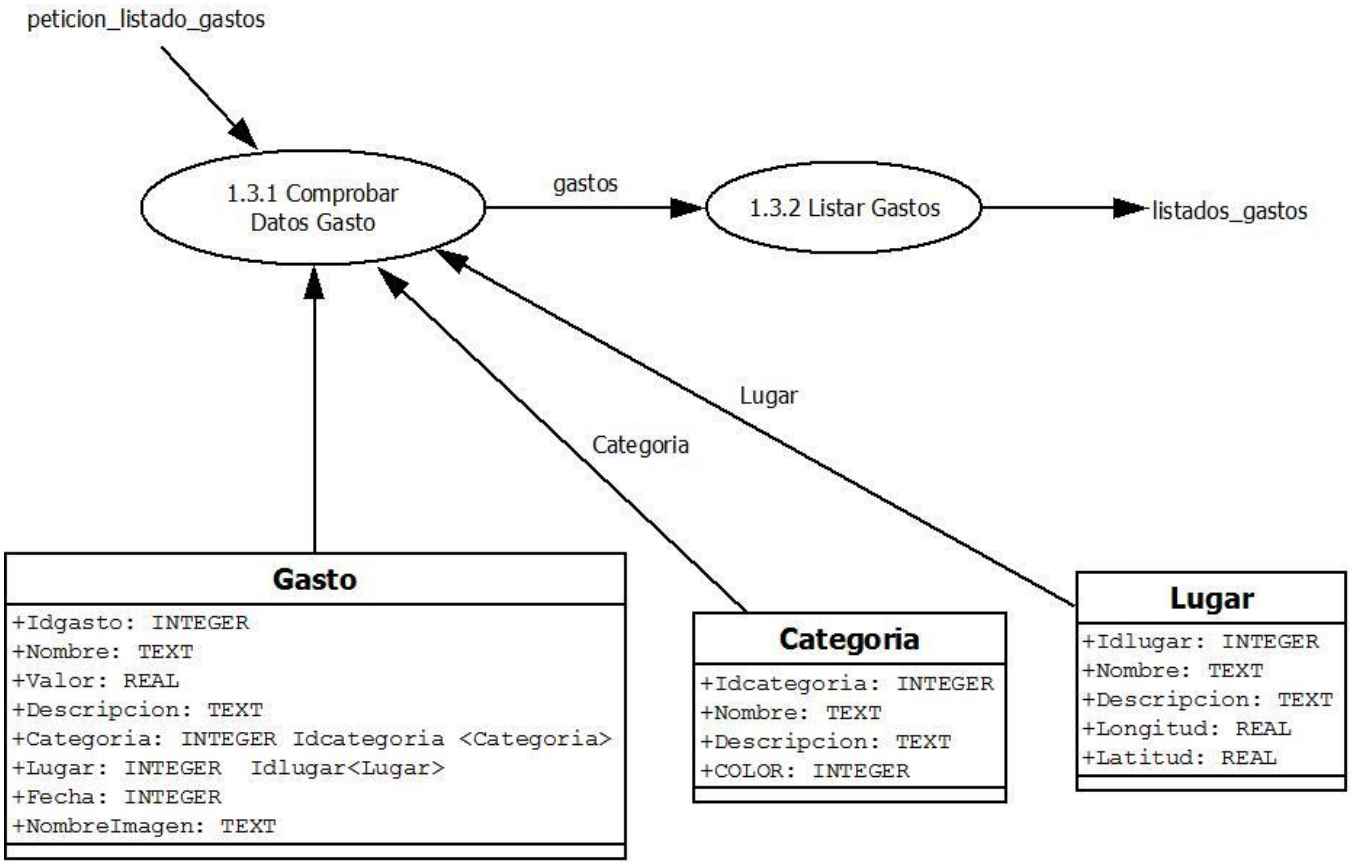
- **E1 GESTIÓN DE UN GASTO (BAJA, MODIFICACIÓN)**



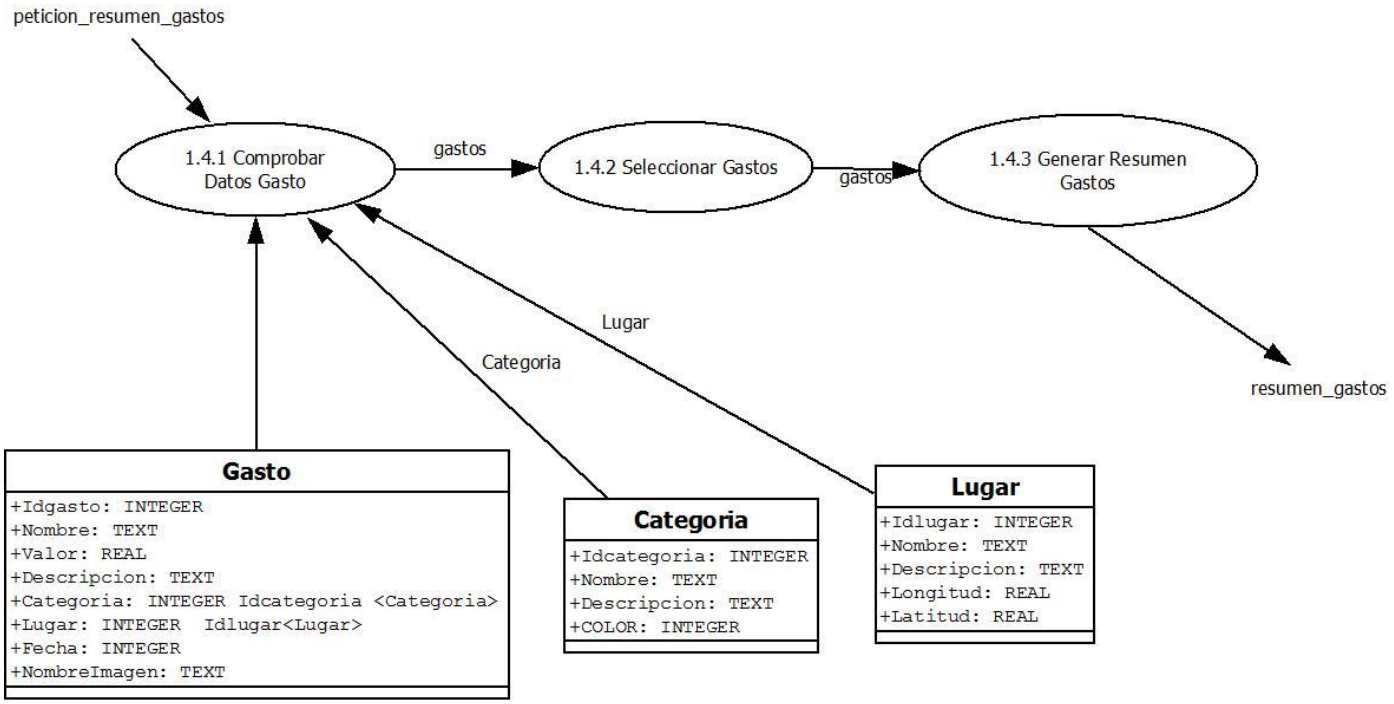
- **E2 ALTA DE UN GASTO**



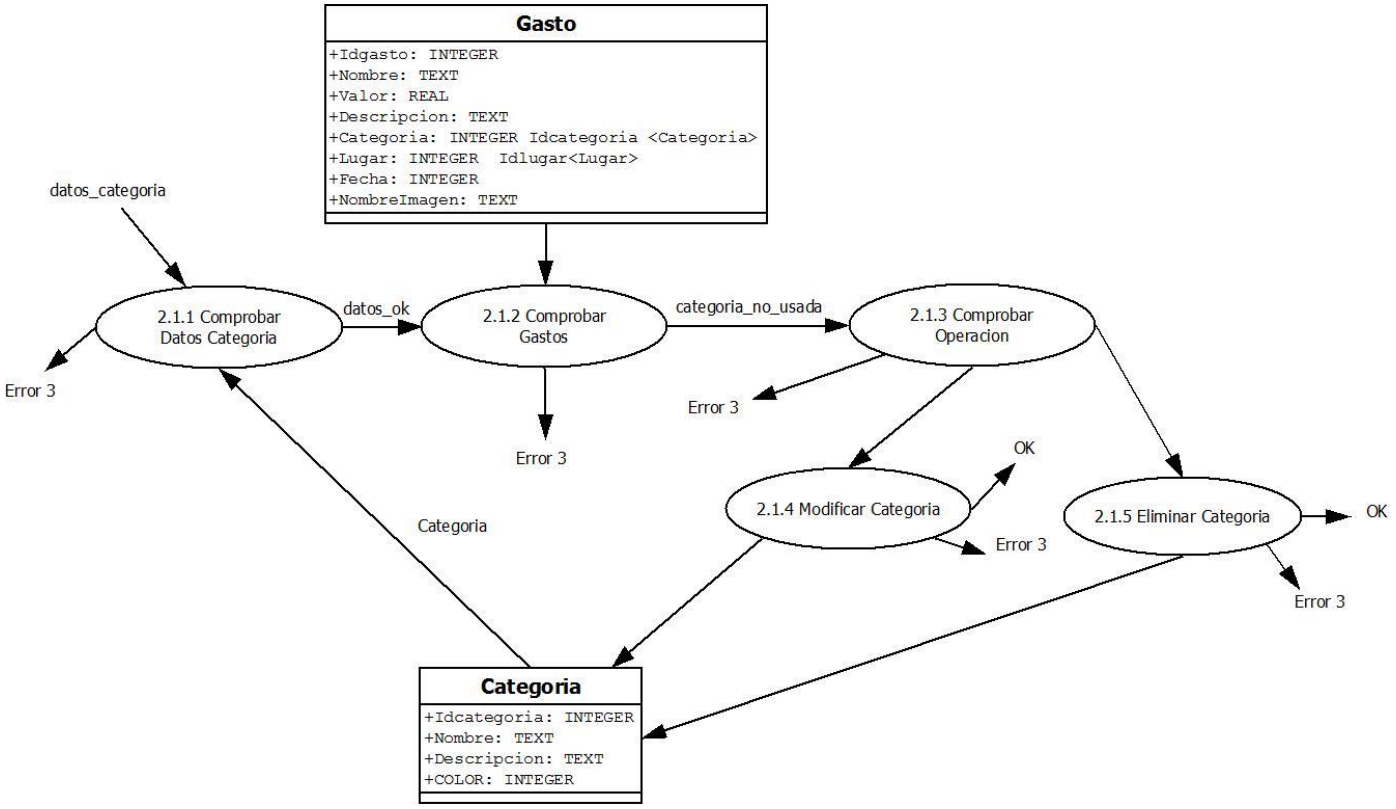
- **E3 GENERAR LISTADO DE GASTOS**



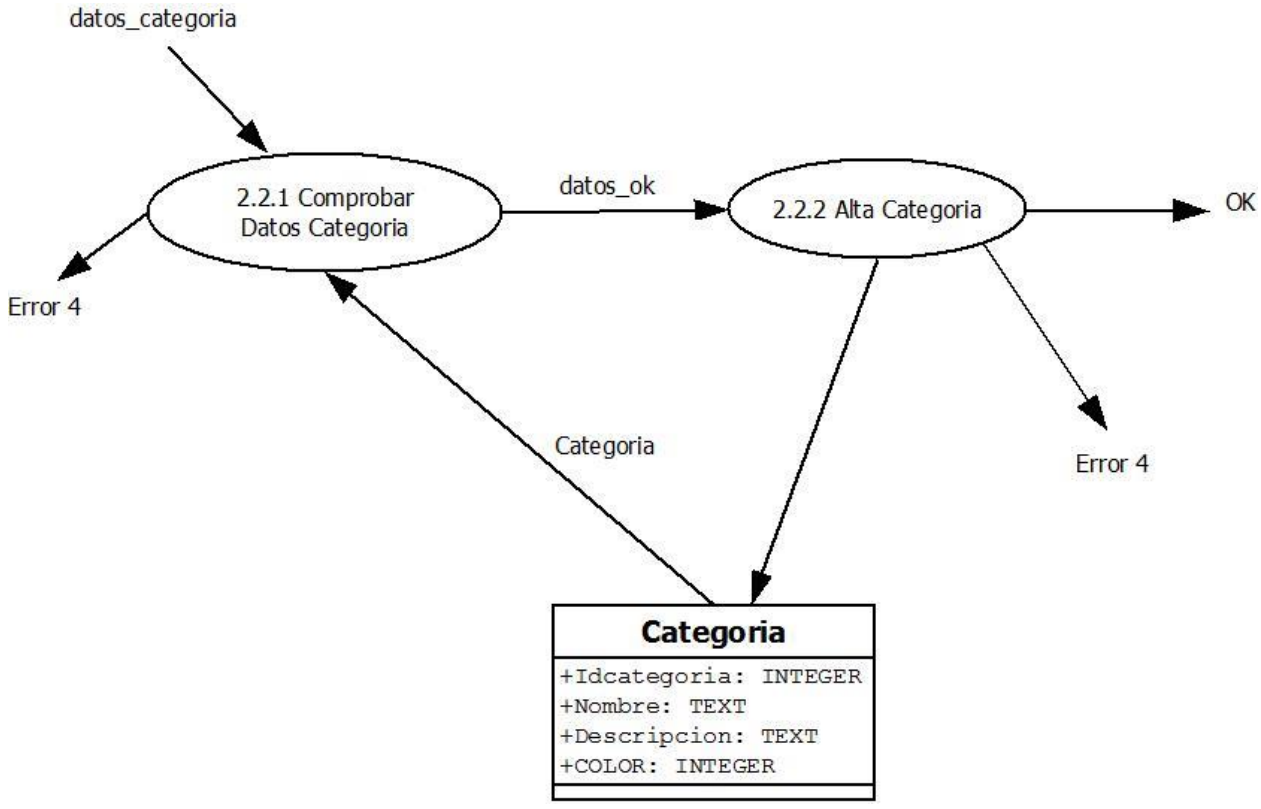
- **E4 GENERAR RESUMEN DE GASTOS**



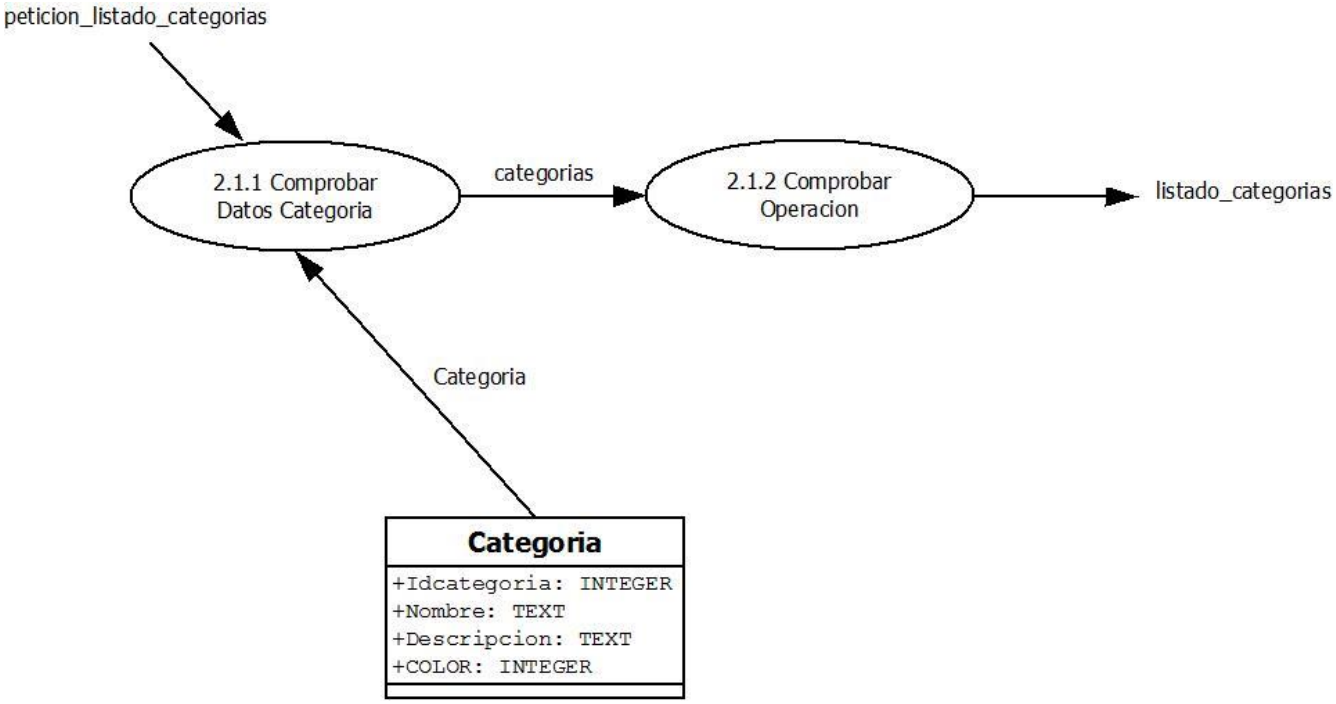
- **E5 GESTIÓN DE UNA CATEGORÍA (BAJA, MODIFICACIÓN)**



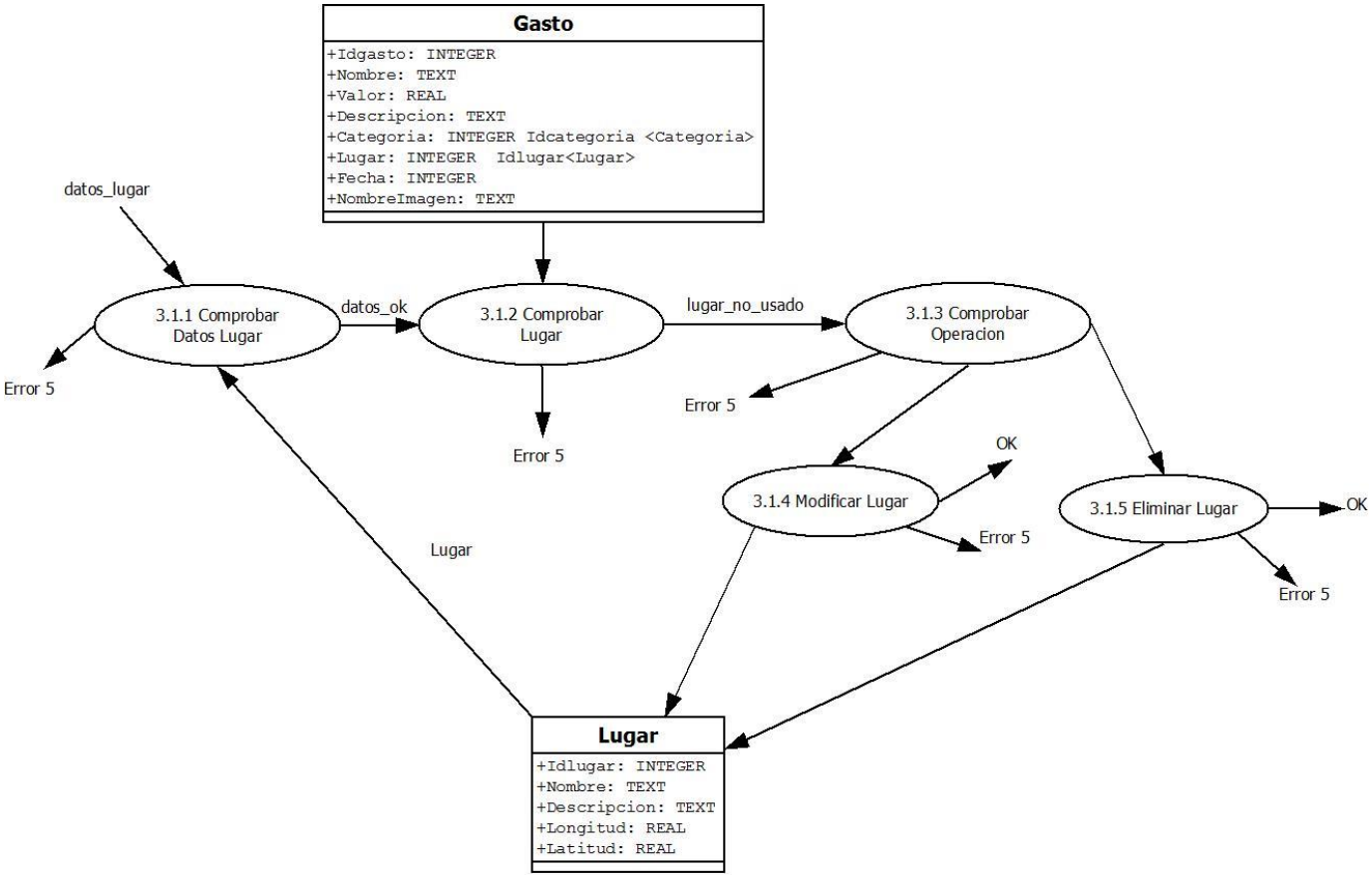
- **E6 ALTA DE UNA CATEGORÍA**



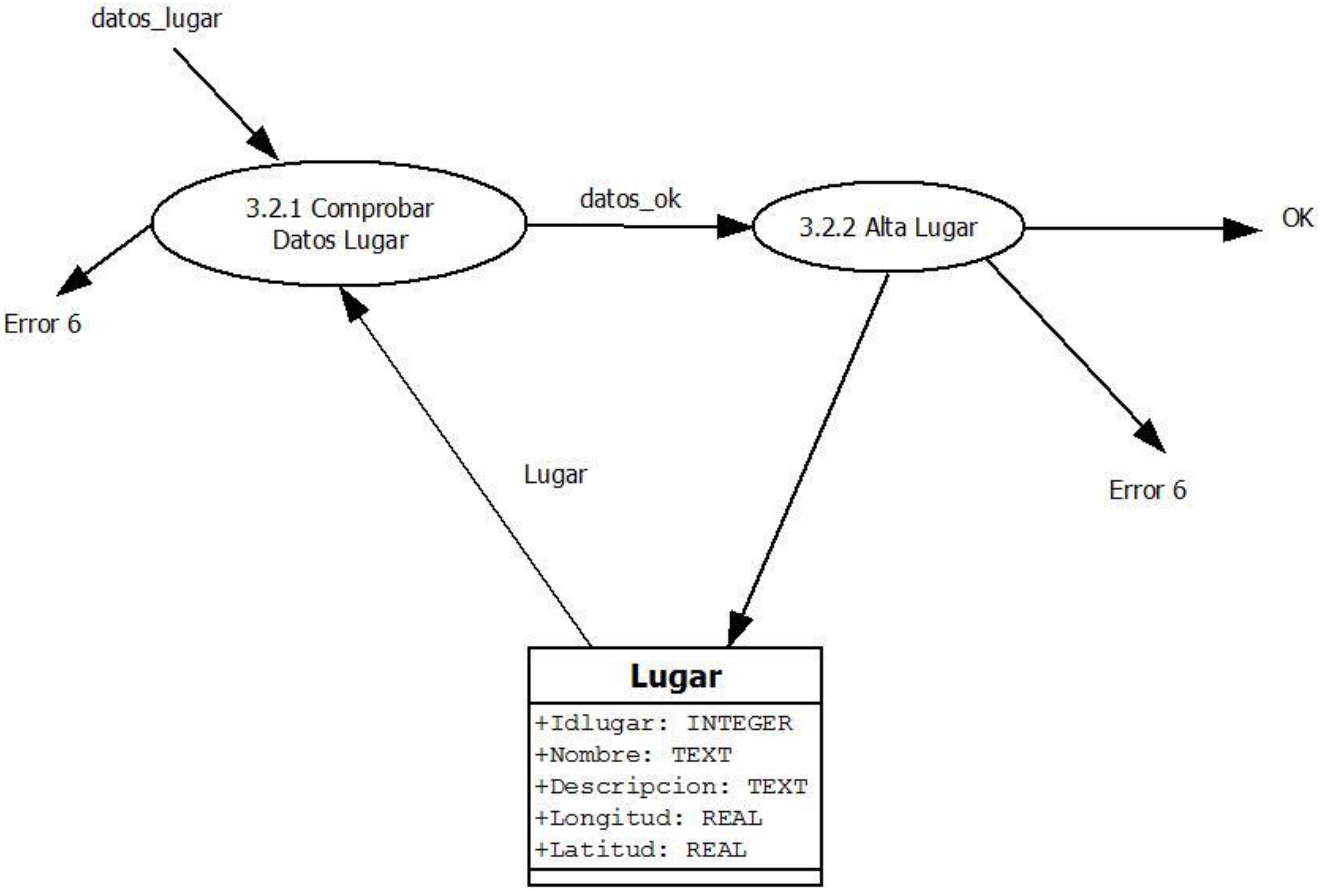
- **E7 GENERAR LISTADO DE CATEGORÍAS**



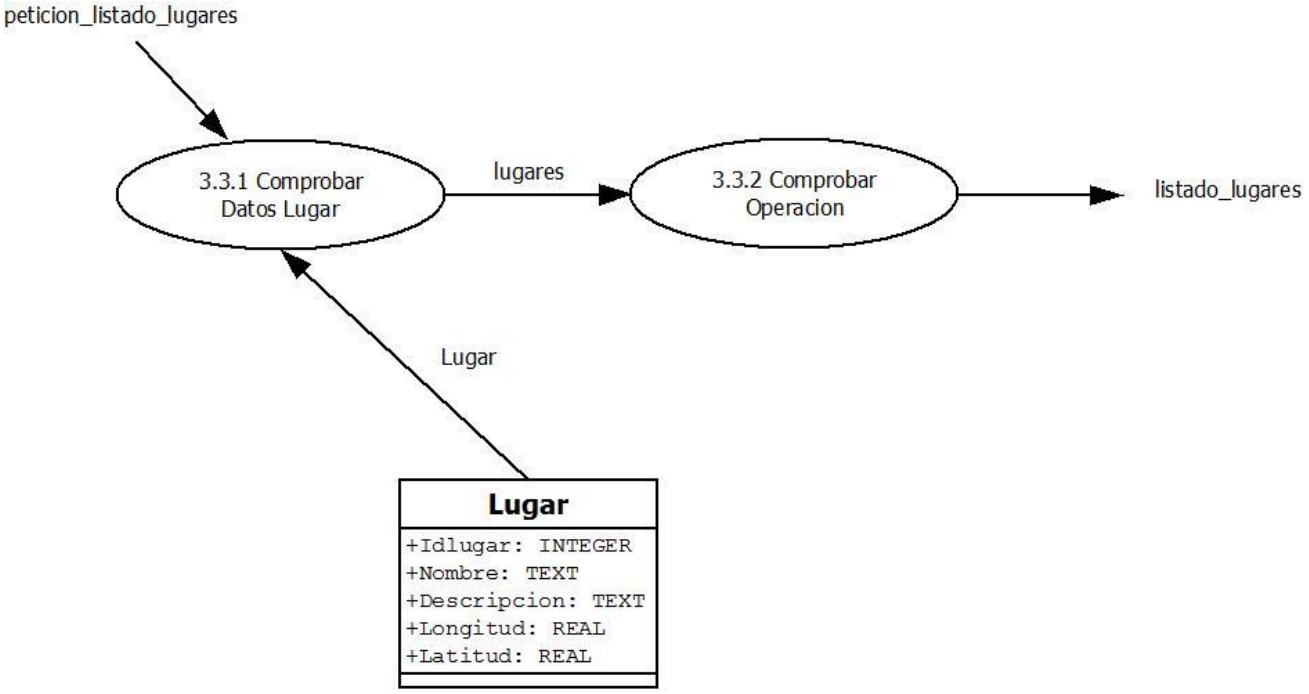
- **E8 GESTIÓN DE UN LUGAR (BAJA, MODIFICACIÓN)**



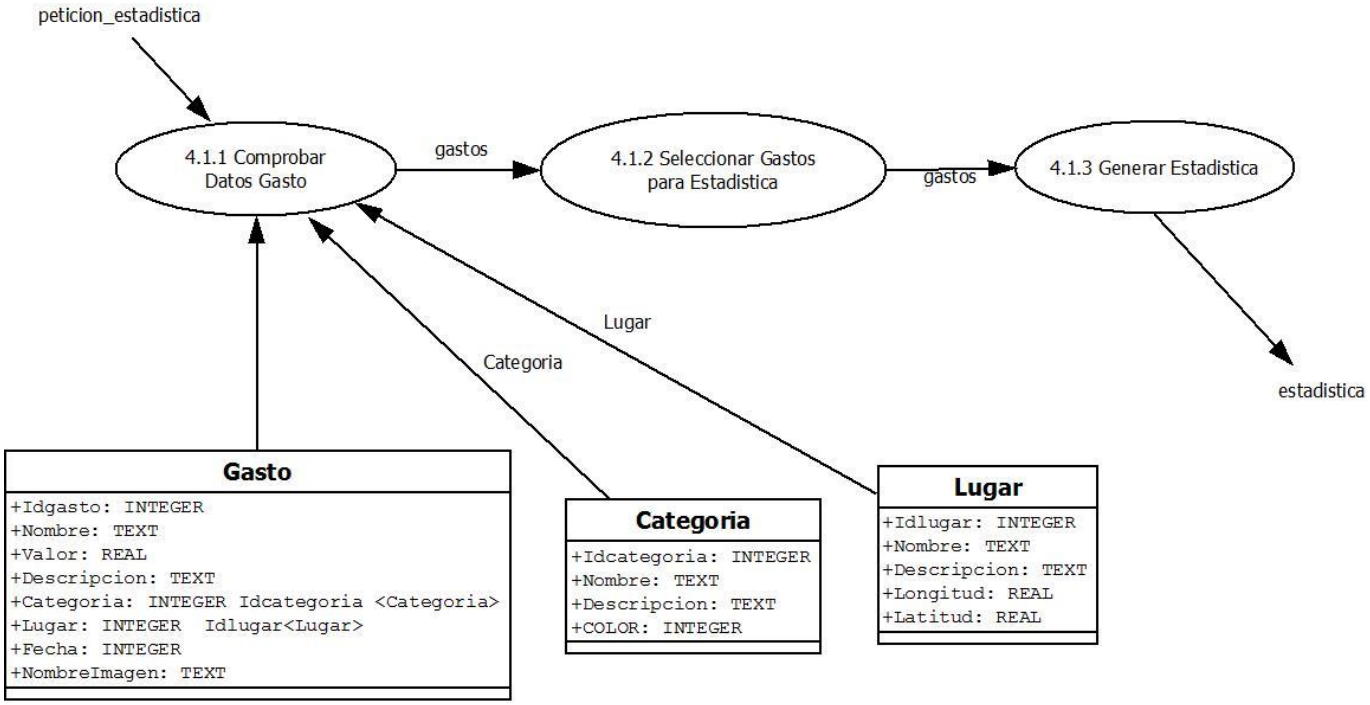
- **E9 ALTA DE UN LUGAR**



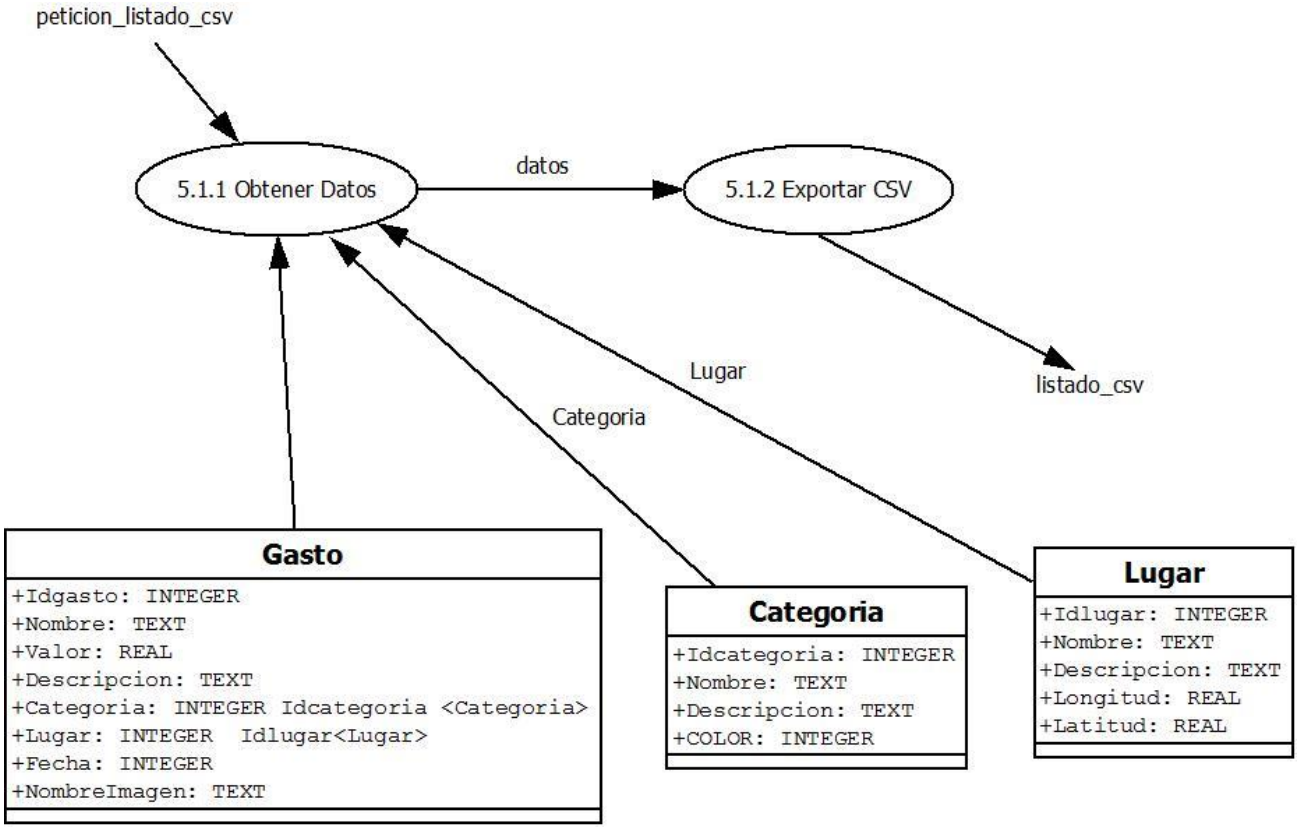
- **E10 GENERAR LISTADO DE LUGARES**



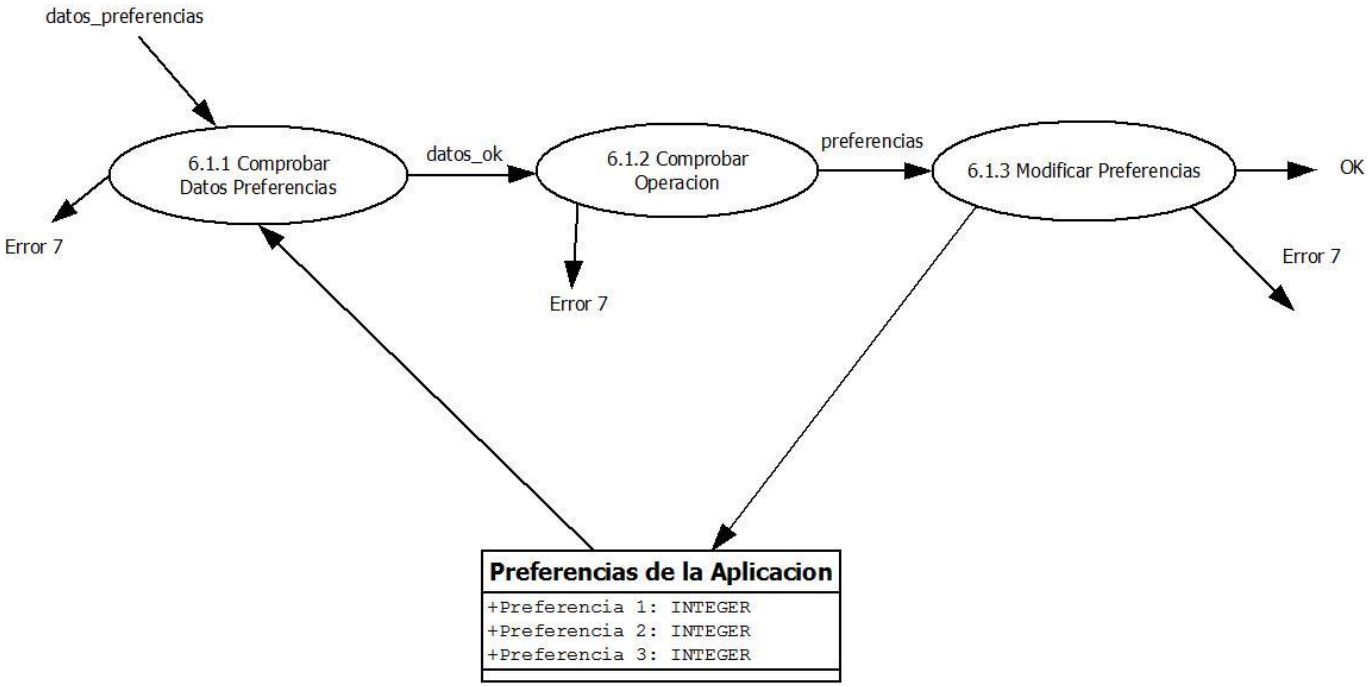
- **E11 GENERAR ESTADÍSTICA**



- **E12 GENERAR DATOS EXPORTACIÓN A CSV.**

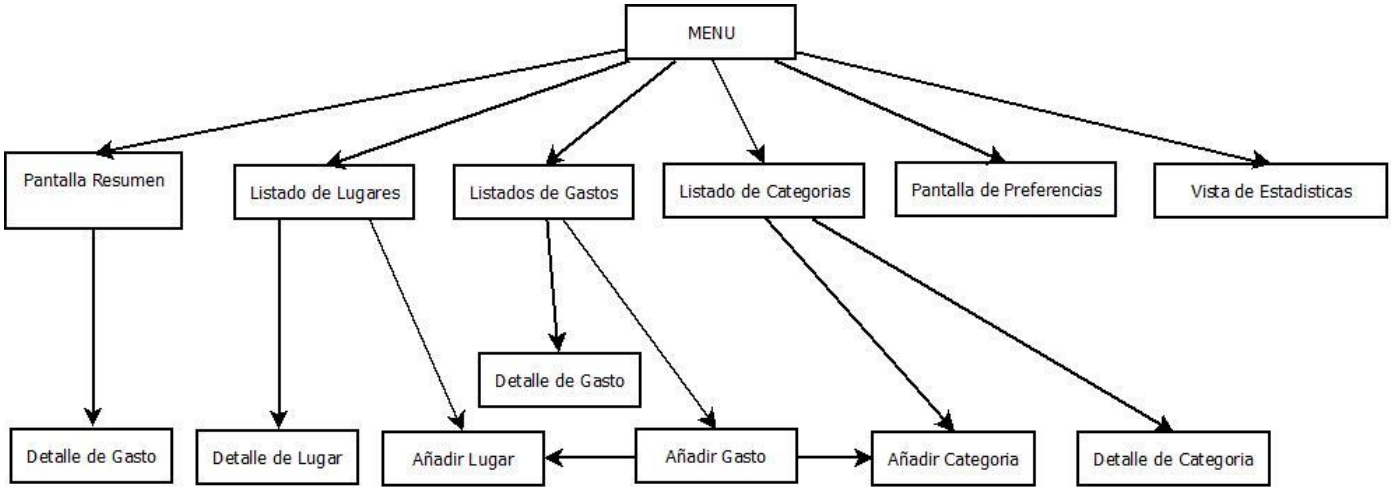


- **E13 GESTIÓN PREFERENCIAS**



4.7 ÁRBOL DE NAVEGACIÓN DE LA APLICACIÓN

A continuación, les mostrare un esquema del árbol de navegación de la aplicación:



Más adelante detallaremos el contenido de cada una de las pantallas, así como la interfaz final desarrollada en la fase de implementación.

Ahora realizare un pequeño resumen del contenido de cada pantalla mostrada en el árbol de navegación de la aplicación:

- **Menú**: Es el menú principal de la aplicación, de esta se podrá acceder a las distintas funcionalidades de la aplicación.
- **Pantalla Resumen**: Se mostrara un resumen de los últimos gastos.
- **Listado de Gastos**: Se mostrara un listado de los gastos de la base de datos.
- **Listado de Lugares:** Se mostrara un listado de los lugares de la base de datos.
- **Listado de Categorías** Se mostrara un listado de las categorías de la base de datos.
- **Pantalla de Preferencias:** Pantalla con las preferencias de la aplicación.
- **Vista de Estadísticas**: Pantalla donde se mostraran las estadísticas.
- **Detalle de Gasto:** Formulario con la información del Gasto.
- **Añadir Gasto**: Formulario para introducir la información del Gasto.
- **Detalle de Lugar:** Formulario con la información del Lugar.
- **Añadir Lugar**: Formulario para introducir la información del Lugar.
- **Detalle de Categoría**: Formulario con la información de la Categoría.
- **Añadir Categoría:** Formulario para introducir la información de la Categoría.

4.8 DISEÑO DE PANTALLAS

PANTALLA MENÚ:

Nombre de pantalla **"MENÚ"**
Opción Pantalla Resumen
Opción Listado de Gastos
Opción Listado de Categorías
Opción Pantalla de Preferencias
Opción Vista de Estadísticas
Opción Añadir nuevo gasto.

La pantalla de Menú será la pantalla principal de la aplicación, desde esta pantalla se podrá acceder a las pantallas de "LISTADO DE GASTOS", "LISTADO DE LUGARES", "LISTADO DE CATEGORIAS", "PANTALLA DE PREFERENCIAS" y "VISTA DE ESTADÍSTICAS".

Desde esta pantalla también se podrá acceder a la opción de añadir un nuevo gasto.

Es la primera pantalla que se mostrara al usuario

PANTALLA RESUMEN:

Nombre de pantalla **"RESUMEN"**
LISTADO ÚLTIMOS GASTOS
Gasto 1 Gasto 2 Gasto . Gasto N
TOTAL GASTOS MENSUALES
TOTAL GASTOS DE HOY
Opción Añadir nuevo gasto.

La pantalla Resumen nos mostrar los últimos gastos que hemos realizado, la cantidad de gastos que se mostraran dependen de una preferencia que el usuario puede modificar.

Además del listado de gastos en esta pantalla se nos mostrara cuantos gastos llevamos en el mes actual, y cuantos gastos en el día actual.

También desde esta pantalla se podrá crear un nuevo Gasto.

LISTADO DE GASTOS:

Nombre de pantalla **"LISTA DE GASTOS"**
Opción Filtrar Gastos
LISTADO GASTOS
Gasto 1 Gasto 2 Gasto . Gasto N
Opción Añadir nuevo Gasto.

En la pantalla listado de gastos, se nos mostraran todos los gastos de la base de datos, el usuario podrá filtrar y ordenar los gastos que aparecen en la pantalla por los diferentes parámetros de los gastos.

Si el usuario selecciona un gasto de la lista, accederá a la pantalla de detalle de gasto, y podrá consultar y modificar sus datos.

También desde esta pantalla se podrá acceder a añadir un nuevo Gasto.

LISTADO DE LUGARES

Nombre de pantalla **"LISTA DE LUGARES"**
Opción Filtrar Lugares
LISTADO LUGARES
Lugar 1 Lugar 2 Lugar . Lugar N
Opción Añadir nuevo Lugar.

En esta pantalla podremos consultar todos los lugares de la base de datos, como en la pantalla de gastos se podrá filtrar y ordenar por los diferentes campos de un Lugar.

Si el usuario selecciona un lugar de la lista, accederá a la pantalla de detalle de lugar, y podrá consultar y modificar sus datos.

Desde esta pantalla se podrá acceder a la opción de crear un nuevo lugar.

LISTADO DE CATEGORIAS

Nombre de pantalla **"LISTA DE CATEGORIAS"**
Opción Filtrar Lugares
LISTADO CATEGORIAS
Categoría 1 Categoría 2 Categoría . Categoría N
Opción Añadir nueva Categoría.

En la pantalla de listado de categorías se podrán consultar todas las categorías de la base de datos, se podrá filtrar y ordenar por los diferentes campos de una categoría.

Si el usuario selecciona una categoría de la lista, accederá a la pantalla de detalle de categoría, y podrá consultar y modificar sus datos.

Desde esta pantalla se podrá acceder a la opción de crear una nueva categoría.

PANTALLA DE PREFERENCIAS

Nombre de pantalla **"PANTALLA DE PREFERENCIAS"**	
Preferencia 1	Valor
Preferencia 2	Valor
Preferencia 3	Valor

En la pantalla de preferencias, al usuario se le mostraran distintas opciones con el valor actual, desde esta pantalla podrá modificar estas preferencias. (Ejemplo de preferencia: Numero de gastos en la pantalla resumen)

VISTA DE ESTADÍSTICAS:

Nombre de pantalla **"VISTA DE ESTADÍSTICAS"**
Opción para seleccionar estadística
Nombre de la estadística
Datos de la estadística.
Grafica de la estadística
Leyenda de la estadística

En esta pantalla se le mostraran distintas estadísticas al usuario, el usuario podrá seleccionar que estadística quieres mostrar, y se le mostrara de forma gráfica y con texto la información.

DETALLE DE GASTO:

Nombre de pantalla **"DETALLE DE GASTO"**		
Nombre:	Valor	
Cantidad:	Valor	
Descripción:	Valor	
Categoría:	Valor	
Lugar:	Valor	
Fecha:	Valor	
Guardar (Botón)	Eliminar (Botón)	Cancelar (Botón)

En esta pantalla al usuario se le mostraran los datos del gasto que haya activado, desde esta pantalla podrá modificar los valores, borrar el gasto de la base de datos, o seleccionar cancelar para anular los cambios.

AÑADIR GASTO:

Nombre de pantalla **"AÑADIR GASTO"**	
Nombre:	Valor
Cantidad:	Valor
Descripción:	Valor
Categoría:	Valor
Lugar:	Valor
Fecha:	Valor
Añadir Categoría (Botón)	Añadir Lugar (Botón)
Guardar (Botón)	Cancelar (Botón)

En esta pantalla el usuario podrá introducir los valores del gasto que quiere crear en la base de datos. También podrá usar el botón Cancelar para salir de la pantalla sin crear un gasto nuevo.

DETALLE DE LUGAR:

Nombre de pantalla **"DETALLE DE LUGAR"**		
Nombre:	Valor	
Descripción:	Valor	
Guardar (Botón)	Eliminar (Botón)	Cancelar (Botón)

En esta pantalla al usuario se le mostraran los datos del lugar que haya seleccionado, y podrá modificar los valores, borrar el lugar de la base de datos, o seleccionar cancelar para anular los cambios.

AÑADIR LUGAR:

Nombre de pantalla **"AÑADIR LUGAR"**	
Nombre:	Valor
Descripción:	Valor
Guardar (Botón)	Cancelar (Botón)

En la pantalla de añadir lugar, al usuario se le mostraran los diferentes campos de un registro de Lugar, el usuario rellenara el formulario, y podrá guardar o cancelar la creación de un nuevo lugar.

DETALLE DE CATEGORÍA:

Nombre de pantalla **"DETALLE DE CATEGORÍA"**		
Nombre:	Valor	
Descripción:	Valor	
Color:	Valor	
Guardar (Botón)	Eliminar (Botón)	Cancelar (Botón)

En esta pantalla al usuario se le mostraran los datos de la categoría que haya seleccionado, y podrá modificar los valores, borrar la categoría de la base de datos, o seleccionar cancelar para anular los cambios

AÑADIR CATEGORÍA:

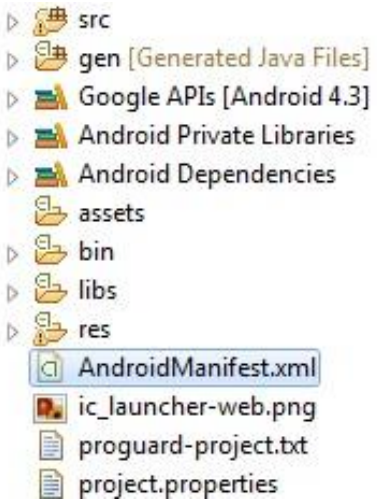
Nombre de pantalla **"AÑADIR CATEGORÍA"**	
Nombre:	Valor
Descripción:	Valor
Color:	Valor
Guardar (Botón)	Cancelar (Botón)

En esta pantalla, al usuario se le mostraran un formulario los diferentes campos de un de una Categoría, el usuario rellenara el formulario, y podrá guardar o cancelar la creación de un nueva categoría.

5. IMPLEMENTACIÓN

5.1 ESTRUCTURA DE LA APLICACIÓN ANDROID

Una aplicación Android está dividida en diferentes ficheros y carpetas, a continuación se explicara para que sirve cada fichero y carpeta:



- **AndroidManifest.xml:** Este fichero define la mayoría de los parámetros de la Aplicación, como son Nombre, permisos necesarios por la aplicación, versión de SDK utilizada, las diferentes actividades por la que está formada la aplicación.

  AndroidManifest.xml del proyecto:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.com.apm.pfcn"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="18" />

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
    <uses-permission android:name="android.permission.READ_LOGS" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="com.example.com.apm.pfcn.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
         android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
```

```
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.com.apm.pfcn.MainActivity"
            android:configChanges="keyboardHidden|orientation"
            android:label="@string/app_name"
            android:parentActivityName="MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".PreferencesActivity"
            android:label="Preferencias" >
        </activity>

        <service
            android:name=".GPSTracker"
            android:enabled="true" >
        </service>

        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyDYnhnUprV4LZp14xhfcGY7TZ3-DinIqIg" />
    </application>
</manifest>
```

- **Carpeta src:** En esta carpeta es donde se almacenan los diferentes ficheros .java del proyecto.
- **Carpeta Google APIs**: En esta carpeta están los diferentes ficheros .jar que representan las Apis de Google que son usadas por la aplicación.
- **Android private libraries:** Aquí se guardan las diferentes librerías externas usadas en la aplicación.
- **Android Dependencies:** Si el proyecto depende de algún otro proyecto se indica aquí, como es el caso de tess-two la librería de reconocimiento óptico de caracteres.
- **Assets:** diferentes ficheros de datos que pueden ser usados por la aplicación.
- **Bin**: ficheros generados a partir de los archivos java de la carpeta src.
- **Libs:** Ficheros de librerías en formato jar usadas en la aplicación.
- **Res:** Aquí están todos los ficheros xml, que definen los strings usados por la aplicación, y los diferentes layouts de las pantallas.

5.2 IMPLEMENTACIÓN DE LA BASE DE DATOS

La implementación de la Base de Datos del Modelo Relacional mostrado anteriormente, se realiza en el fichero BDGastos.java.

Se implementa la clase BDHelper que extiende la clase SQLiteOpenHelper que es la clase destinada en Android como interfaz entre la base de datos y la aplicación.

Primero se implementa la función **public void** onCreate(SQLiteDatabase db), esta función se encarga de crear las diferentes tablas de la base de datos.

```
public void onCreate(SQLiteDatabase db) {

        // Creamos tabla de CATEGORIA
        db.execSQL("CREATE TABLE " + N_TABLA2 + "(" + ID_FILA2 + " INTEGER PRIMARY KEY
        AUTOINCREMENT, " + ID_NOMBRE2 + " TEXT NOT NULL, " + ID_DESCRIPCION2 + " TEXT NOT
        NULL, " + ID_COLOR2 + " INTEGER);");

        // Creamos tabla de LUGAR
        db.execSQL("CREATE TABLE " + N_TABLA3 + "(" + ID_FILA3 + " INTEGER PRIMARY KEY
        AUTOINCREMENT, " + ID_NOMBRE3 + " TEXT NOT NULL, " + ID_DESCRIPCION3 + " TEXT NOT
        NULL, " + ID_LATITUD3 + " REAL, " + ID_LONGITUD3 + " REAL);");

        // creamos tabla de GASTOS
        db.execSQL("CREATE TABLE " + N_TABLA + "(" + ID_FILA + " INTEGER PRIMARY KEY
        AUTOINCREMENT, " + ID_NOMBRE      + " TEXT NOT NULL, " + ID_VALOR + " REAL NOT
        NULL, " + ID_DESCRIPCION + " TEXT NOT NULL, " + ID_CATEGORIA + " INTEGER, " +
        ID_LUGAR + " INTEGER," + ID_FECHA + " INTEGER, " + ID_FOTO + " TEXT);");

        // Creamos tabla de IDIOMA
        db.execSQL("CREATE TABLE " + N_TABLA4 + "(" + ID_FILA4 + " INTEGER PRIMARY KEY
        AUTOINCREMENT, " + ID_NOMBRE4 + " TEXT NOT NULL, " + ID_DIRECCION4 + " TEXT NOT
        NULL, " + ID_FICHEROCOMP4 + " TEXT NOT NULL, " + ID_FICHERO4 + " TEXT NOT NULL, "
        + ID_INSTALADO4 + " INTEGER NOT NULL);");

}
```

Después hemos de definir las funciones destinadas a abrir y cerrar la Base de Datos, que son:

```
// Función para abrir la BD y poder realizar consultas.
        public BDGastos abrir() {
              nHelper = new BDHelper(nContexto);
              nBD = nHelper.getWritableDatabase();
              return this;
        }
```

```
// Función para cerrar la BD
        public void cerrar() {
              nHelper.close();
        }
```

Ahora ya se pueden implementar las diferentes funciones, para añadir, modificar, y eliminar gastos, categorías, lugares e idiomas a la Base de Datos.

Como ejemplo veremos las funciones relacionadas con gastos:

```
// Función para añadir un Gasto nuevo a la BD
public long crearGasto(String nombre, double valor, String Descripción, Integer
        Categoría, Integer Lugar, long Fecha, String foto) throws SQLException {
        ContentValues cv = new ContentValues();
        cv.put(ID_NOMBRE, nombre);
        cv.put(ID_VALOR, valor);
        cv.put(ID_DESCRIPCION, Descripción);
        cv.put(ID_CATEGORIA, Categoría);
        cv.put(ID_LUGAR, Lugar);
        cv.put(ID_FECHA, Fecha);
        cv.put(ID_FOTO, foto);
        return nBD.insert(N_TABLA, null, cv);
}
```

```
// Función para editar un Gasto de la BD
        public void editarGasto(long fila, String nombre, double valor, String
        Descripción, Integer Categoría, Integer Lugar,
                    long Fecha, String foto) throws SQLException {
              ContentValues cv = new ContentValues();
              cv.put(ID_NOMBRE, nombre);
              cv.put(ID_VALOR, valor);
              cv.put(ID_DESCRIPCION, Descripción);
              cv.put(ID_CATEGORIA, Categoría);
              cv.put(ID_LUGAR, Lugar);
              cv.put(ID_FECHA, Fecha);
              cv.put(ID_FOTO, foto);
              nBD.update(N_TABLA, cv, ID_FILA + "=" + fila, null);

        }
```

```
// Función para Eliminar un Gasto de la BD
public void eliminarGasto(long fila) throws SQLException {
        nBD.delete(N_TABLA, ID_FILA + "=" + fila, null);
}
```

Una vez añadida información a la Base de Datos se pueden realizar consultas, como ejemplo podemos ver las siguientes consultas:

Función getCategoria, esta función sirve para obtener un objeto categoría a partir del identificador de categoría.

Se realizar una query de la base de datos a partir del identificador pasado en la función, y a partir de los datos obtenido se crea un nuevo objeto categoría y se devuelve este.

```
public Categoría getCategoria(long rowId) throws SQLException {
        nHelper = new BDHelper(nContexto);
        nBD = nHelper.getWritableDatabase();
        Cursor mCursor = nBD.query(true, N_TABLA2, new String[] { ID_FILA2, ID_NOMBRE2,
        ID_DESCRIPCION2, ID_COLOR2 }, ID_FILA2+ "=" + rowId, null, null, null, null,
        null);
        if (mCursor != null) {
              mCursor.moveToFirst();
```

```
        }
        Categoría categoria = new Categoría(mCursor.getInt(0),mCursor.getString(1),
        mCursor.getString(2), mCursor.getInt(3));
        nHelper.close();
        return categoría;
}
```

Función gastoMaximo, esta función se encarga de devolver el gasto máximo dentro de un periodo de tiempo.

```
public double GastoMaximo(double inicio, double fin) {
        double cantidad = 0;
        Cursor c = nBD.rawQuery("select MAX(Valor) from Gastos WHERE Fecha BETWEEN " +
        inicio + " AND " + fin + ";", null);
        if (c.moveToFirst())cantidad = c.getDouble(0);
        else cantidad = -1;
        c.close();
        return cantidad;
}
```

Función getColorCategoria, esta función devuelve el color asignado a una categoría

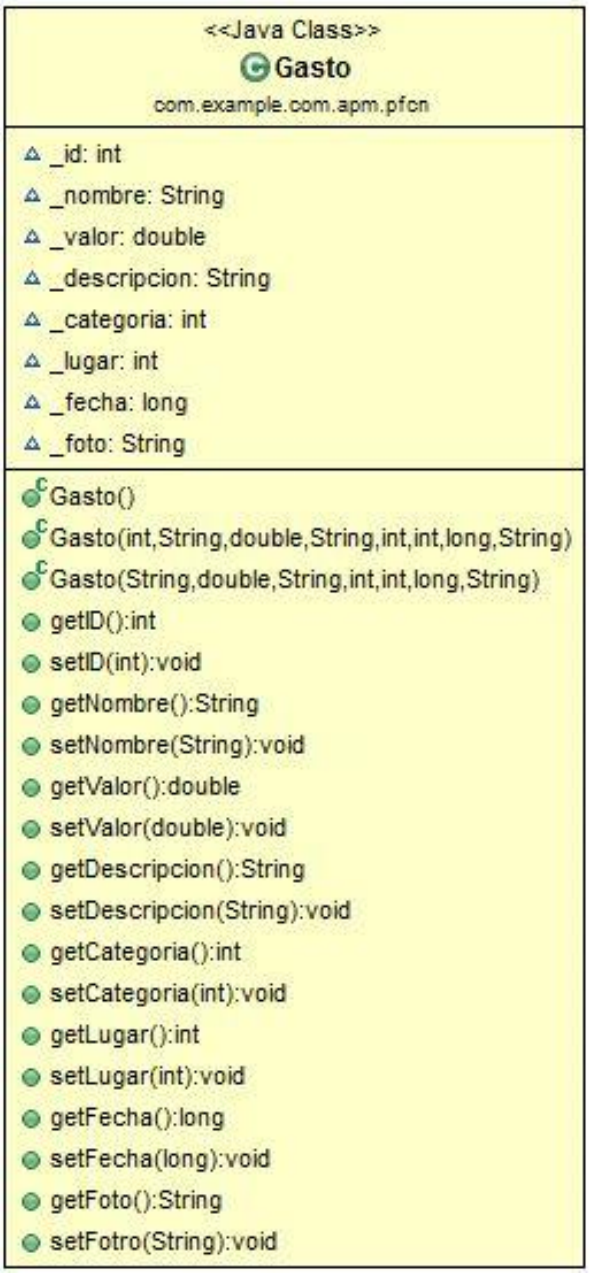
```
public int getColorCategoria(long rowId) throws SQLException {
        nHelper = new BDHelper(nContexto);
        nBD = nHelper.getWritableDatabase();
        Cursor mCursor = nBD.query(true, N_TABLA2, new String[] { ID_COLOR2 }, ID_FILA2
        + "=" + rowId, null, null, null, null,null);
        if (mCursor != null) {
              mCursor.moveToFirst();
        }
        nHelper.close();
        return mCursor.getInt(0);
}
```

Y para finalizar un ejemplo de la **función obtenerColorGastosPorCategoria**, que utiliza la función getColorCategoria mostrada anteriormente:

```
public List<Integer> obtenerColorGastosPorCategoria() {
        Cursor temp = null;
        int idcategoria = 0;
        List<Integer> Resultado = new ArrayList<Integer>();
        temp = nBD.query(N_TABLA2, new String[] { ID_FILA2 }, null, null, null, null,
        null);
        if (temp.moveToFirst()) {
              do {
                    idcategoria = temp.getInt(0);
                    try {
                    Resultado.add(getColorCategoria(idcategoria));
                    } catch (SQLException e) {
                          e.printStackTrace();
                    }
              } while (temp.moveToNext());
        temp.close();
        return Resultado;
}
```

5.3 CLASES

5.3.1 CLASE GASTO:

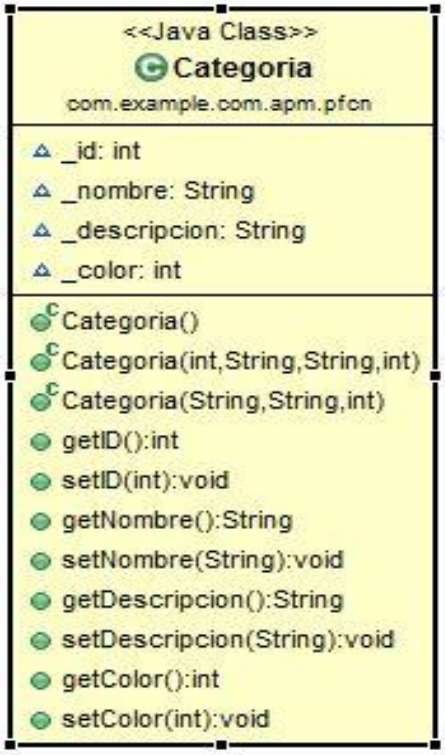


La clase Gasto guarda la información necesaria para definir un objeto gasto.

Podemos destacar que el valor de gasto está guardado con el tipo Double, y la fecha en un tipo Long, que representa a la fecha en milisegundos.

Para cada atributo de la clase tiene una función para obtener y asignar un valor.

5.3.2 CLASE CATEGORÍA:

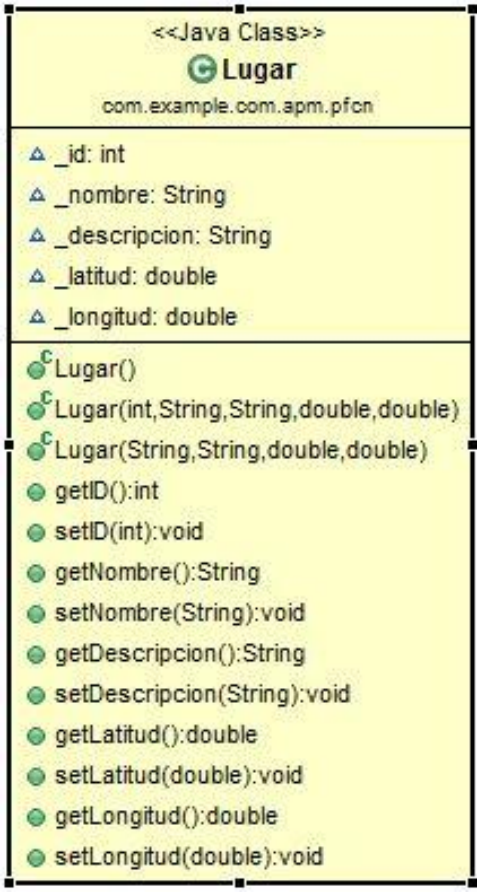


La clase Categoría es el equivalente de la clase gasto pero para el objeto categoría.

Color es un tipo Int, y el resto de atributos son Strings.

Para cada atributo de la clase tiene una función para obtener y asignar un valor.

5.3.3 CLASE LUGAR:



La clase lugar define el objeto lugar con sus atributos y funciones.

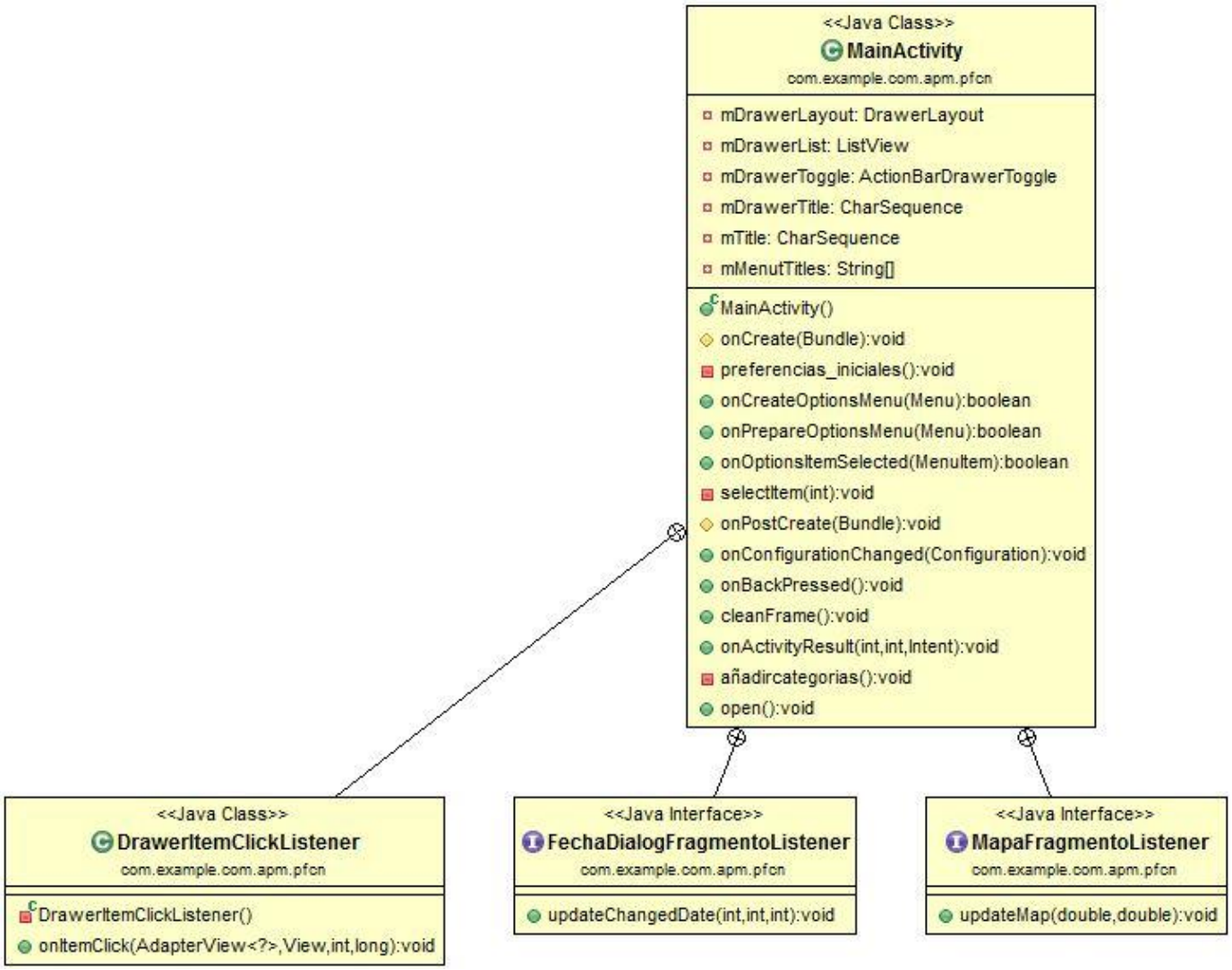
Como podemos ver se guarda la latitud y longitud en un tipo Double, que es el tipo que devuelven las funciones relacionadas con obtener la posición.

Para cada atributo de la clase tiene una función para obtener y asignar un valor.

5.3.4 ACTIVIDADES Y FRAGMENTOS DE LA APLICACIÓN

5.3.4.1 MAINACTIVITY:

Es la actividad principal de la aplicación:



Esta clase extiende la clase FragmentActivity, que es el equivalente en fragments de la clase Activity.

Como toda clase de Android se divide en diferentes partes que se ejecutan dependiendo del estado de la Actividad (ver ciclo de vida de una aplicación Android).

A continuación resumiremos que hace MainActivity en cada uno de estos estados:

• **onCreate:**

- Obtiene los diferentes objetos su layout, y asigna un título a la barra de acción.
- Procede a cargar las preferencias iniciales de la aplicación.
- Define los diferentes eventos de la navegación lateral de la aplicación
- Añade las categorías por defecto de la aplicación
- Carga el fragmento por defecto, que en el caso de la aplicación es el fragmento resumen.

- **onCreateOptionsMenu:**
  - Crea y asigna el menú de la aplicación definido en el fichero xml, menú.xml.
- **onPrepareOptionsMenu:**
  - Si la navegación lateral está abierta oculta el Menú de la aplicación
- **onOptionsItemSelected:**
  - En esta función se define la lógica del menú de la aplicación.
  - Dependiendo de la opción seleccionada mediante un switch se ejecuta la opción adecuada.
- **DrawerItemClickListener:**
  - Define el evento para controlar que opción se ha seleccionado en el menú lateral.
  - Llama a la función selectItem con el número de posición del menú lateral seleccionado.
- **selectItem:**
  - Mediante un switch del valor de posición del menú lateral elegimos que pantalla mostraremos.
  - En el caso de nuestra aplicación elegiremos que fragmento ejecutaremos.
  - Para lanzar un fragmento se ha de realizar el siguiente código:

```
//creamos un fragmento
Fragment fragment = new NOMBREFRAGMENTO();
//Creamos los argumentos que pasaremos al fragmento
Bundle args = new Bundle();
//Asignamos los argumentos al fragmento
fragment.setArguments(args);
//Obtenemos el fragmentmanager que se encarga de gestionar //fragmentos
FragmentManager fragmentManager = getSupportFragmentManager();
//Se realiza la transacción
fragmentManager.beginTransaction().replace(R.id.content_frame, fragment,
        "Main").commit();
```

- **public void onBackPressed():**
  - Controlamos la salida de la aplicación, antes de salir de la aplicación se vuelve a la pantalla de resumen, y si está ya en la pantalla de resumen se muestra un dialogo de confirmación para salir.
- **FechaDialogFragmentoListener:**
  - Es el interfaz que une el fragmento para elegir una fecha con la aplicación:

```
public interface FechaDialogFragmentoListener {
// Este interfaz es un listener entre el fragmento FechaDialogo
// Fragmento y la actividad
     public void updateChangedDate(int year, int month, int day);
}
```

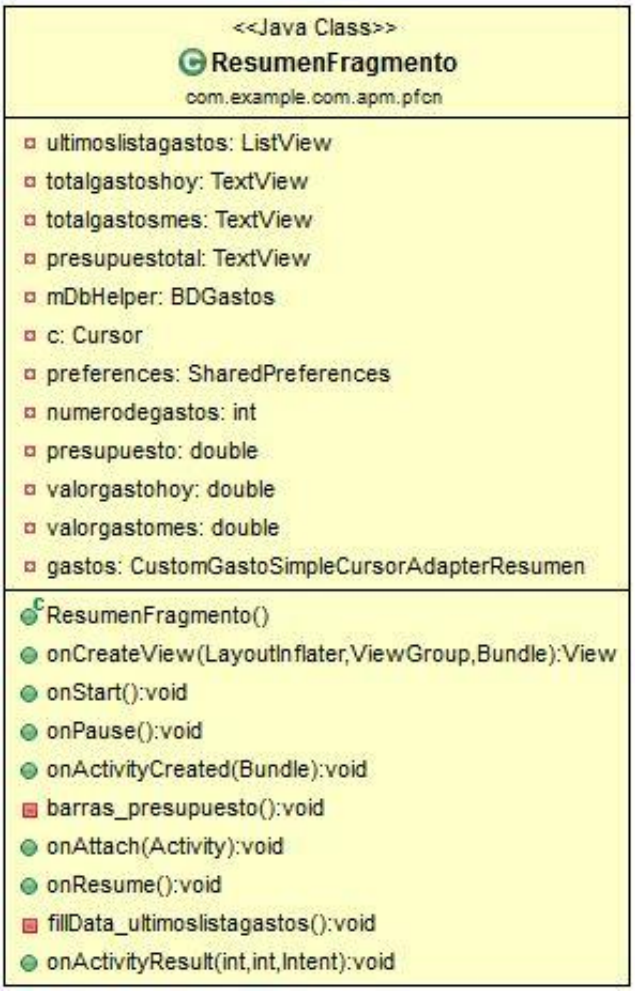
- **MapaFragmentoListener:**
  - Es el interfaz que une el fragmento para elegir una posición por mapa con la aplicación.

```
public interface MapaFragmentoListener {
        // Este interfaz es un listener entre el fragmento MapaFragmento y la
        actividad
    public void updateMap(double nuevalatitud, doublé nuevalongitud);
}
```

- **onActivityResult:**
  - Gestionamos el resultado de las diversas actividades externas que lanzamos desde nuestra actividad.
- **Añadircategorias:**
  - En esta función llamamos a la función de BDGastos que crea las categorías por defecto de la aplicación.

5.3.4.2 FRAGMENTOS FUNCIONALIDADES DE GASTOS:

- **RESUMEN FRAGMENTO:**



Este fragmente es el primero que se le muestra al usuario al iniciar la aplicación.

Contiene la funcionalidad destinada a mostrar los últimos gastos del usuario, e información sobre los últimos gastos del usuario.

- **LISTADO GASTO FRAGMENTO:**

Este fragmento, muestra al usuario una lista con todos los gastos de la base de datos. Usa 4 Spinners para ordenar y filtrar por diferentes campos el listado de gastos. Aquí tenemos un ejemplo de definición de unos de los spinners:

```
// Definimos el evento de cambio de campo en el spinner
spinnerorden.setOnItemSelectedListener(new OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parentView, View selectedItemView,
        int position, long id) {
        campoorden = spinnerorden.getItemAtPosition(position).toString();
        switch (position) {
              case 0:
                    campoorden = "g._id";
                    break;
              case 1:
                    campoorden = "Nombre";
                     break;
              case 2:
                     campoorden = "Valor";
                     break;
              case 3:
                     campoorden = "Descripción";
                     break;
              case 4:
                     campoorden = "NombreCategoria";
                     break;
              case 5:
```

```
                        campoorden = "NombreLugar";
                        break;
                case 6:
                        campoorden = "Fecha";
                        break;
                default:
                        campoorden = "_id";
                        break;
                }
        ActualizarListaGastos(listagastos, c, gastos, campoorden, direccionorden,
        idfiltro, filtro);
        }
        @Override
        public void onNothingSelected(AdapterView<?> parentView) {
        }
});
```

La función más importante de este fragmento es ActualizarListaGastos, esta función se encarga de actualizar el listview con el listado de gastos dependiendo de los valores de los spinners.

```
protected void ActualizarListaGastos(ListView lista, Cursor cursor,
        CustomGastoSimpleCursorAdapterResumen adapter, String orden, String
        direccion, long id, String stringfiltro) {
        try {
              cursor = mDbHelper.fetchAllGastos(orden, direccion, id, stringfiltro);
        } catch (SQLException e) {
              e.printStackTrace();
        }
        // Definimos los datos que mostraremos en el Listview
        String[] from = new String[] { BDGastos.ID_NOMBRE, DGastos.ID_NOMBRE3,
        BDGastos.ID_VALOR };
        int[] to = new int[] { R.id.text1, R.id.text2, R.id.text6 };
        // Creamos el adaptador que unirá el cursor de la base de
        // datos con el Listview
        adapter = new CustomGastoSimpleCursorAdapterResumen(getActivity(),
        R.layout.gastos_row_nuevo, cursor, from, to);
        // Asignamos el adaptador al Listview
        lista.setAdapter(adapter);
        // indicamos al adaptador que hemos cambiado el cursor
        adapter.changeCursor(cursor);
}
```

- **NUEVO GASTO FRAGMENTO:**

Este fragmento es el formulario usado para dar de alta un nuevo gasto en la Base de Datos.

Una función que podemos destacar de este fragmento es TomarFoto, es la función de llamar a la aplicación Galería del sistema operativo Android, e indicarle que queremos tomar una foto.

Ejemplo de función TomarFoto():

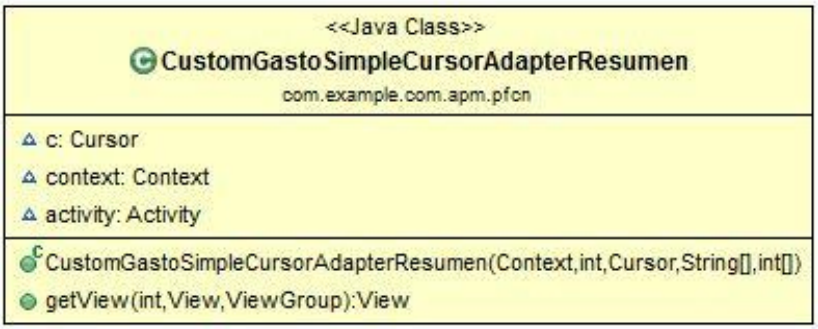
```
public void TomarFoto() {
        Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
              Calendar ahora = Calendar.getInstance();
        String nombreImagen = "Gasto " + Long.toString(ahora.getTimeInMillis()) +
        ".jpg";
        File photo = new File(Environment.getExternalStorageDirectory(),
        nombreImagen);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photo));
        uriImagen = Uri.fromFile(photo);
        getActivity().startActivityForResult(intent, 100);
}
```

- **DETALLE GASTO FRAGMENTO**

Detalle gasto fragmento es muy similar a Nuevo gasto fragmento, es el fragmento destinado para editar y borrar un gasto de la base de datos.

Las principales diferentes funcionalidades, es que a este fragmento se le pasa mediante un argumento el identificador del gasto que vamos a modificar, y tiene más lógica para obtener los datos de este gasto, y poder modificarlos.

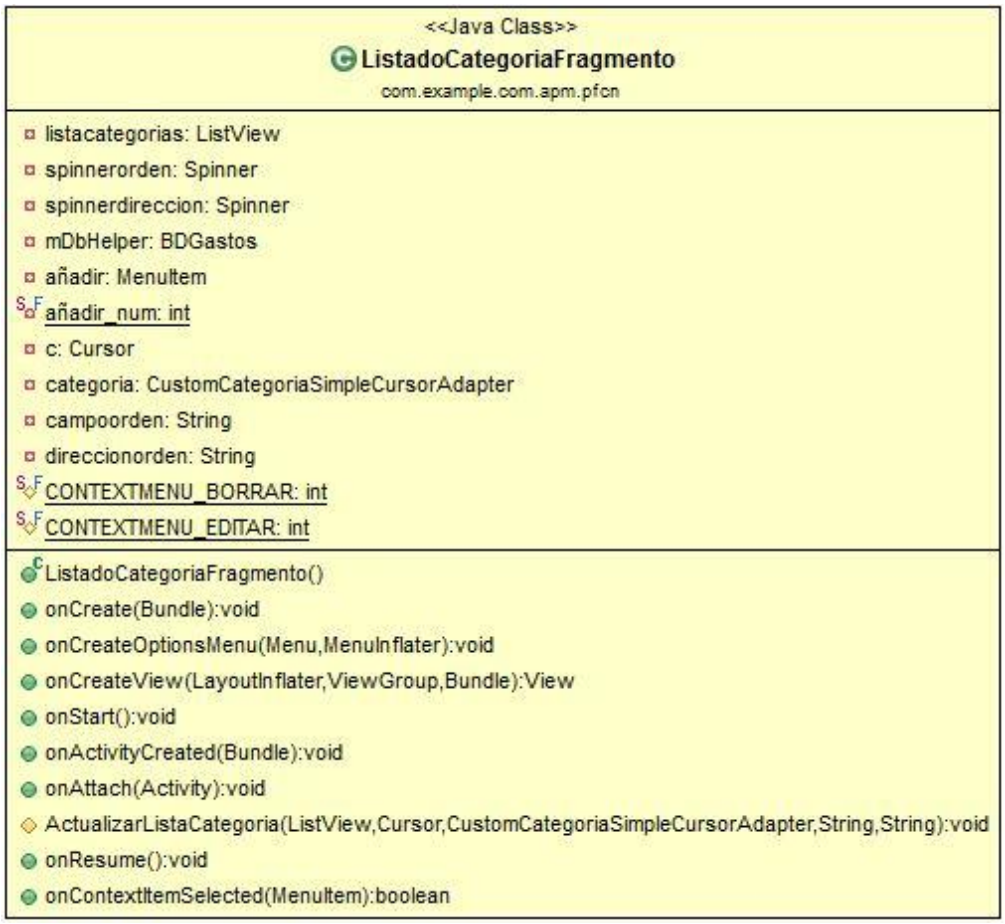
- **CUSTOM GASTO SIMPLE CURSOR ADAPTER**



Esta clase es una herencia de la clase cursor adapter, se usa para cambiar el aspecto del listview donde mostramos los gastos.

5.3.4.3 FRAGMENTOS FUNCIONALIDADES DE CATEGORÍA:

- **LISTADO CATEGORÍAS FRAGMENTO**



Este es otro fragmento destinado a mostrar las categorías presentes en la base de datos.

La principal funcionalidad de este fragmento está en los spinners destinados a filtrar y ordenador las categorías, y la función ActualizarListaCategorias() que se encarga de actualizar el listview por los valores asignados por los spinners.

También se puede mediante una pulsación larga de una de las categorías mostrar un menú contextual para borrar o modificar la categoría. Aquí podemos ver como se define este evento:

```
listacategorias.setOnCreateContextMenuListener(new OnCreateContextMenuListener() {

@Override
        public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo
        menuInfo) {
              menu.setHeaderTitle("Menu");
              menu.add(0, CONTEXTMENU_EDITAR, 0, "Editar");
              menu.add(0, CONTEXTMENU_BORRAR, 1, "Borrar");
        }

});
```

- **NUEVO CATEGORÍA FRAGMENTO**

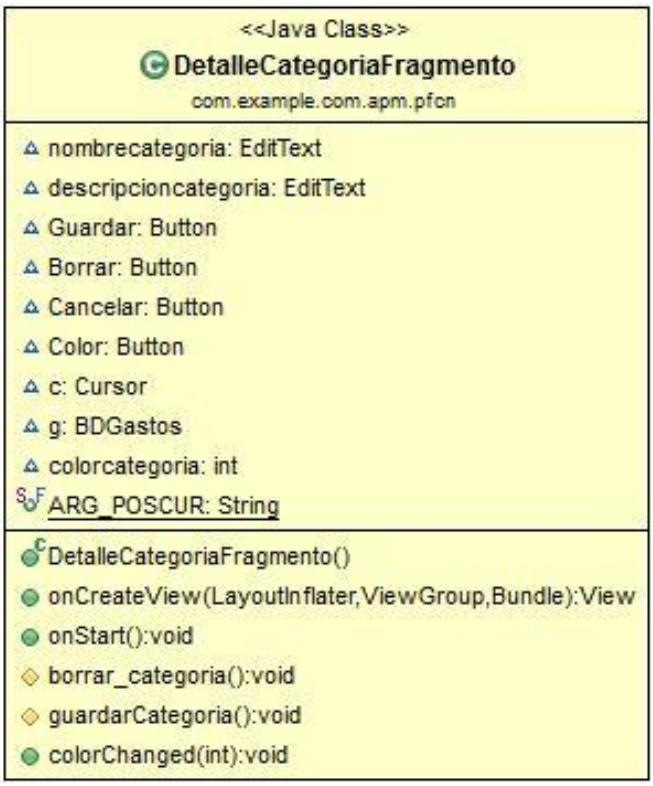
Es el fragmento que implementa el formulario para crear una nueva Categoría en la Base de Datos.

Aquí podemos ver el proceso que se realiza para crear una nueva categoría:

```
private void crear_categoria() {
        Boolean error = false;
        cat = new Categoría();
        // Obtenemos los valores del formulario
        cat.setDescripcion(descripcioncategoria.getText().toString());
        cat.setNombre(nombrecategoria.getText().toString());
        cat.setColor(colorcategoria);
        // Comprobamos que el nombre de la categoría sea correcto
        if (Soporte.EstaVacio(nombrecategoria)) {
              AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
              builder.setTitle("Nombre de Categoría Vacío");
              builder.setMessage("No se puede crear una Categoría sin Nombre");
              builder.setPositiveButton("OK", null);
              AlertDialog dialog = builder.show();
              error = true;
        }
        // Si no hay error creamos en la BD la Categoría con los
        // nuevos valores
        if (error == false) {
              BDGastos nuevo = new BDGastos(getActivity());
              try {
                    // Abrimos la BD
                    nuevo.abrir();
              } catch (Exception e) {
                    e.printStackTrace();
              }
              try {
                    // Creamos la Categoría con los nuevos valores
                    nuevo.crearCategoria(cat.getNombre(), cat.getDescripcion(),
        cat.getColor());
              } catch (SQLException e) {
                   e.printStackTrace();
              }
              // Cerramos la BD
              nuevo.cerrar();
```

```
                // Volvemos al fragmento anterior
                getActivity().getSupportFragmentManager().popBackStack();
        }
}
```

- **DETALLE CATEGORÍA FRAGMENTO:**



Este fragmento es utilizado para borrar o modificar los valores de una categoría. A este fragmento como parámetro se le pasa el id de la categoría que tenemos que modificar o borrar.

Aquí podemos ver un ejemplo de la función borrar_categoria, que es usada para eliminar una categoría de la base de datos:

```
protected void borrar_categoria() {
        BDGastos nuevo = new BDGastos(getActivity());
        Categoría categoria = null;
        try {
                categoria = g.getCategoria(getArguments().getLong(ARG_POSCUR));
        } catch (SQLException e1) {
                e1.printStackTrace();
        }
        // Comprobamos que la categoría no esté en uso por un Gasto
        if (g.Categoriaenuso(categoria.getID())) {
        // Mostramos un dialogo de error si la categoría está en uso
        // y cancelamos la acción
                AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
                builder.setTitle("Error al borrar Categoría");
                builder.setMessage("No se puede borrar una Categoría si está en uso por un
        Gasto");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.show();
                TextView messageView = (TextView)
        dialog.findViewById(android.R.id.message);
                messageView.setGravity(Gravity.CENTER);
        } else {
```

```
            // Abrimos la BD para borrar la Categoría
            try {
                  nuevo.abrir();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            // Eliminamos la Categoría
            try {
            nuevo.eliminarCategoria(getArguments().getLong(ARG_POSCUR));
            } catch (SQLException e) {
                  e.printStackTrace();
            }
            // Cerramos la BD
            nuevo.cerrar();
            // Volvemos al fragmento anterior
            getActivity().getSupportFragmentManager().popBackStack();
           }
      }
```

5.3.4.4 FRAGMENTOS FUNCIONALIDADES DE LUGAR:

- **LISTADO DE LUGARES FRAGMENTO:**

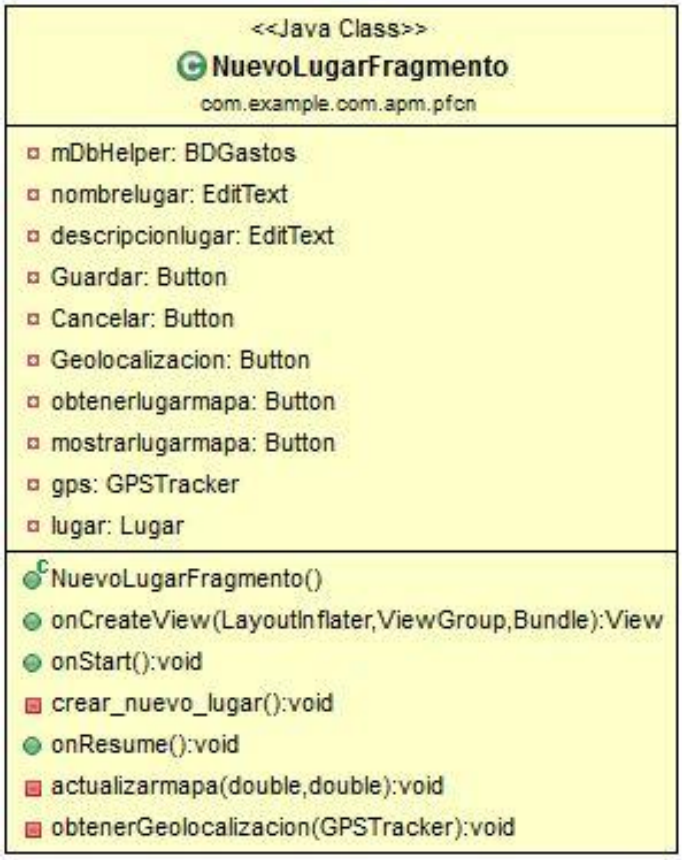
Es el fragmento encargado de mostrar los lugares de la base de datos, como los otros listados de la aplicación se puede ordenar y filtrar.

Aquí podemos ver un ejemplo de cómo se llama al fragmento detalle lugar fragmento apretando sobre cualquier lugar de la lista:

```
listalugares.setOnItemClickListener(new OnItemClickListener() {

		@Override
		public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3)
	{
		// aquí llamamos a fragment_detalle_lugar con la información
		// necesaria en un bundle
		if (arg3 != -1) {
			Fragment newFragment = new DetalleLugarFragmento();
			Bundle args = new Bundle();
			args.putLong(DetalleLugarFragmento.ARG_POSCUR, arg3);
			newFragment.setArguments(args);
			FragmentTransaction transaction =
	getFragmentManager().beginTransaction();
```

```
                    transaction.replace(R.id.content_frame, newFragment);
                    transaction.addToBackStack(null);
                    // Ejecutamos el fragmento
                    transaction.commit();
                }
            }
});
```

- **NUEVO LUGAR FRAGMENTO:**



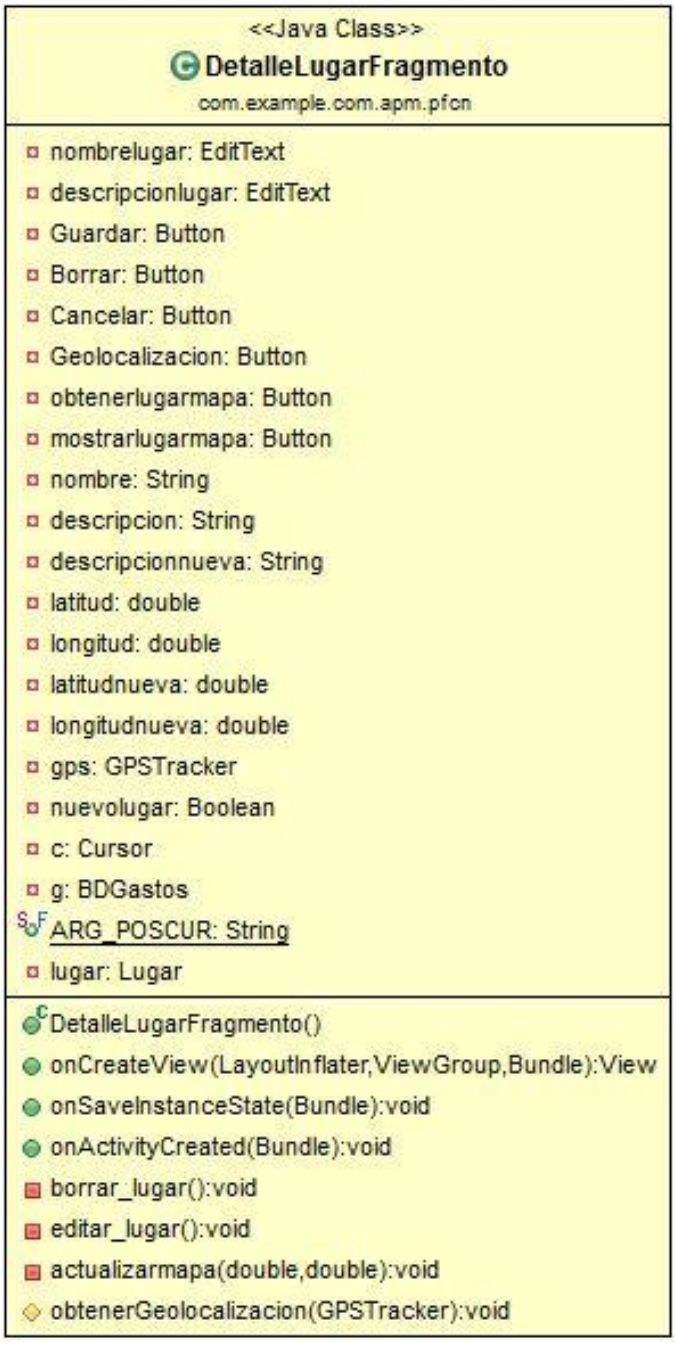
Nuevo lugar fragmento es el fragmento usado para crear un nuevo lugar en la base de datos.

Como ejemplo de este fragmento podemos ver como se lanza el fragmento de mapa para obtener la posición del lugar mediante este fragmento:

```
// Definimos el evento Onclick del Botón ObtenerLugar en el Mapa
obtenerlugarmapa.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View arg0) {
        // Creamos un nuevo fragmento de Mapa y Definimos un
        // Listener para obtener el resultado del mapa con la latitud,
        // longitud y descripcion
        Fragment newFragment = MapaFragmento.newInstance(getActivity(), new
        MapaFragmentoListener() {
              public void updateMap(double nuevalatitud, double nuevalongitud) {
                    actualizarmapa(nuevalatitud, nuevalongitud);
               });
        Bundle args = new Bundle();
        newFragment.setArguments(args);
        args.putDouble(MapaFragmento.ARG_LATITUD, lugar.getLatitud());
        args.putDouble(MapaFragmento.ARG_LONGITUD, lugar.getLongitud());
```

```
        args.putInt(MapaFragmento.ARG_MAPA, 0);
        FragmentTransaction transaction = getFragmentManager().beginTransaction();
        transaction.replace(R.id.content_frame, newFragment);
        transaction.addToBackStack(null);
        // Commit the transaction
        transaction.commit();
        }
});
```

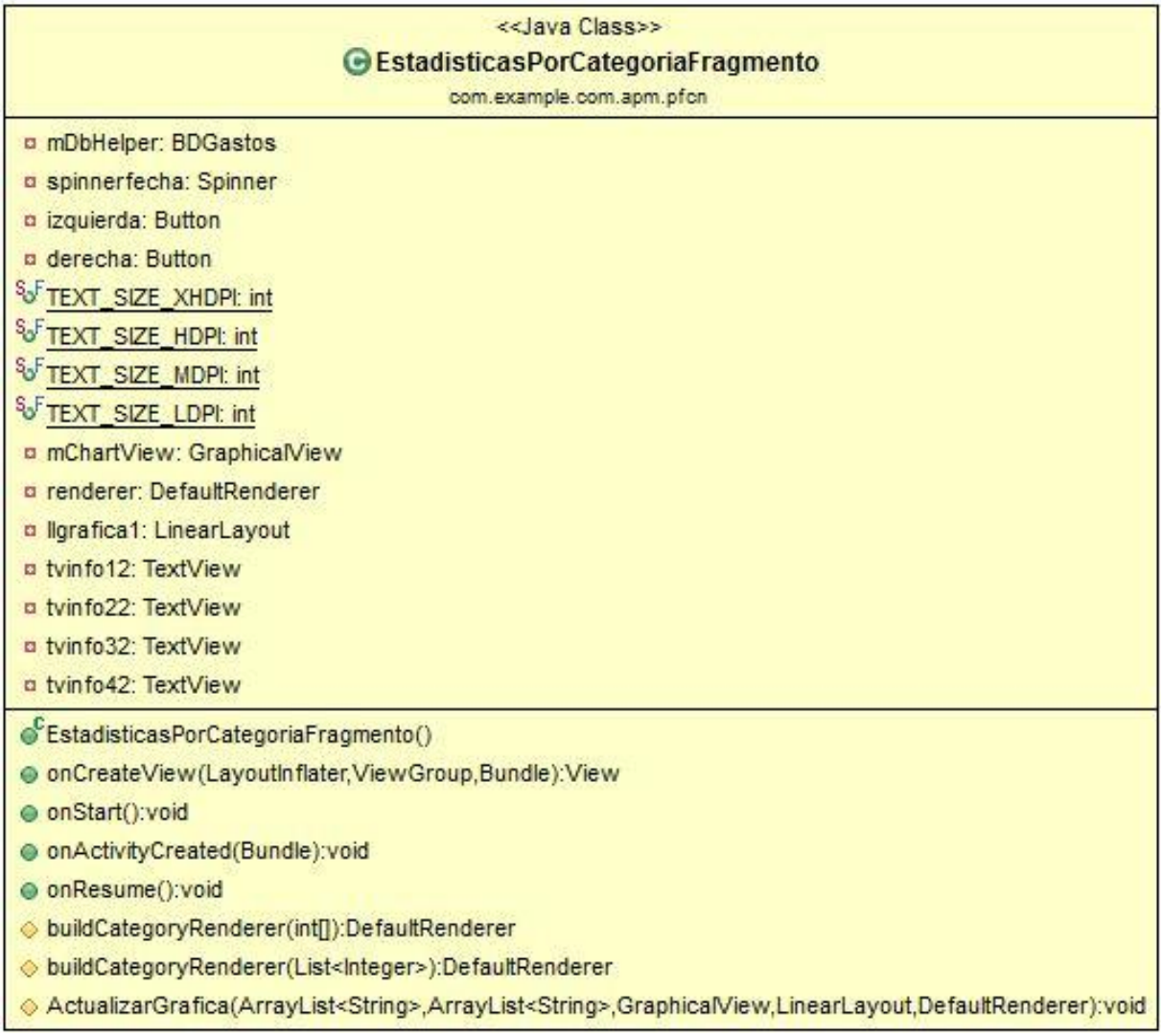
- **DETALLE LUGAR FRAGMENTO:**



Es el fragmento usado para modificar los datos de un lugar. Este fragmento es similar a nuevo lugar fragmento, pero añade bastantes funcionalidades que requieres una clase nueva.

5.3.4.5 FRAGMENTOS FUNCIONALIDADES ESTADÍSTICAS

- **ESTADÍSTICAS POR CATEGORÍA FRAGMENTO:**



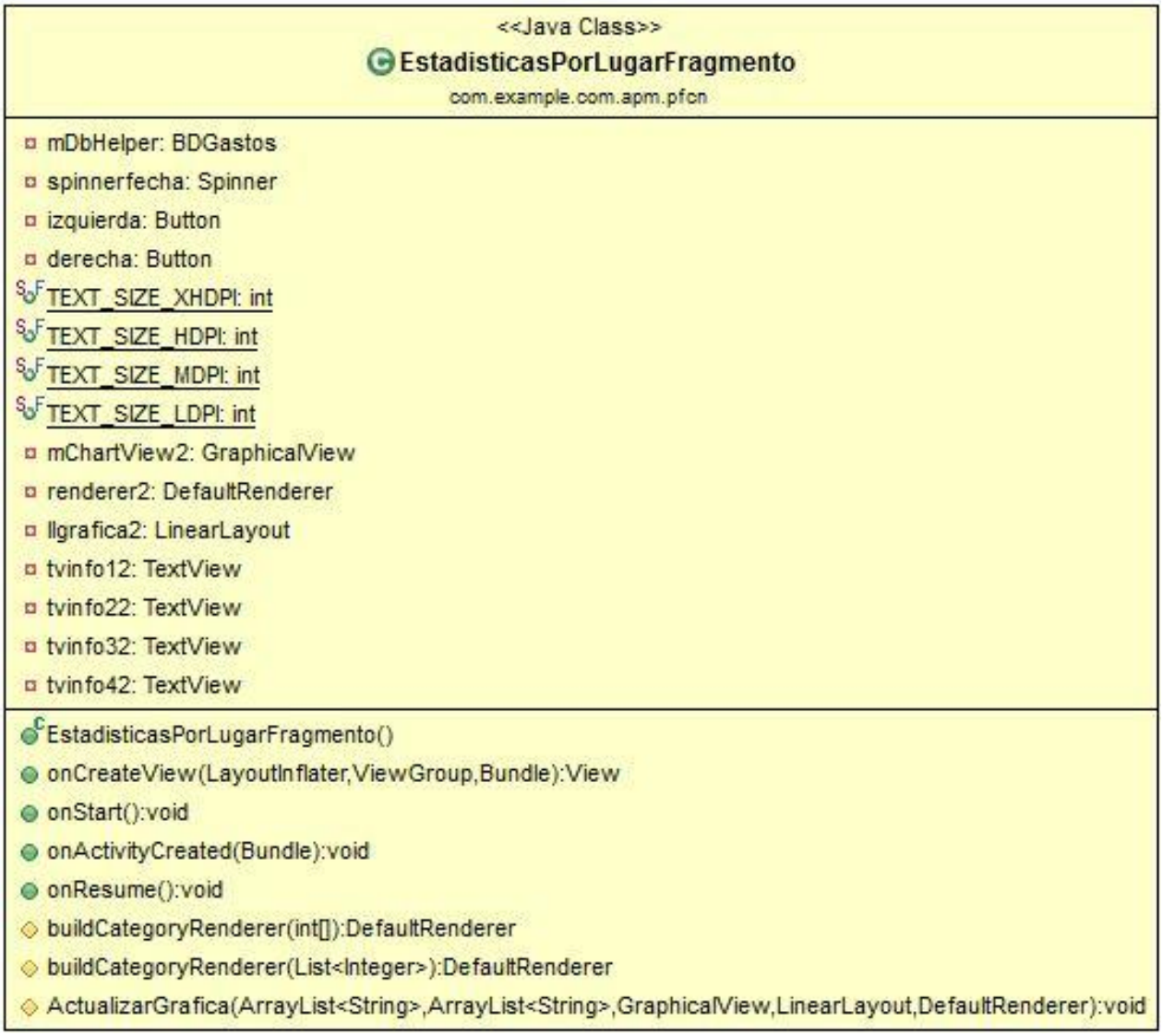
Este fragmento es parte de los fragmentos de la aplicación destinados a mostrar estadísticas al usuario.

Este fragmento muestra un pie chart, con la información de los gastos en diferentes periodos de tiempos por cada una de las categorías.

Un ejemplo de código de este fragmento, esta función se encarga de actualizar la gráfica:

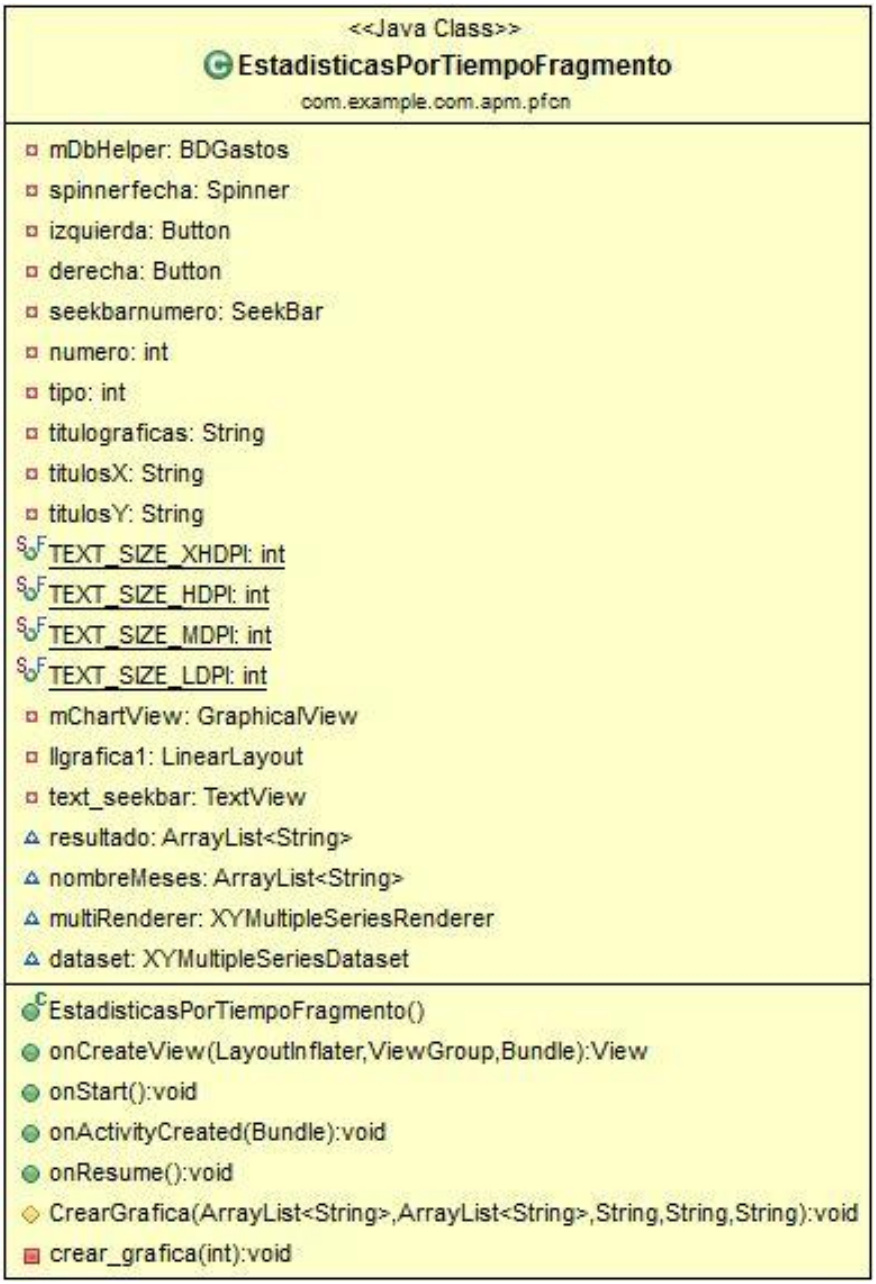
```
protected void ActualizarGrafica(ArrayList<String> resultado, ArrayList<String> nombre,
         GraphicalView mChartV,
         LinearLayout layout, DefaultRenderer render) {
         CategorySeries serie = new CategorySeries("Gastos");
         int listSize;
         listSize = resultado.size();
         for (int i = 0; i < listSize; i++) {
               serie.add(nombre.get(i), Double.valueOf(resultado.get(i)));
         }
         mChartV = ChartFactory.getPieChartView(getActivity(), serie, render);
         mChartV.repaint();
         layout.removeAllViews();
         layout.addView(mChartV);
}
```

- **ESTADÍSTICAS POR LUGAR FRAGMENTO:**



Es un fragmento equivalente a "Estadísticas por categoría fragmento", realiza las mismas funciones pero respecto a los lugares de la base de datos.

- **ESTADÍSTICAS POR TIEMPO FRAGMENTO:**



Este fragmento muestra una gráfica de barras con información de los gastos mostrada en diferentes periodos de tiempos.

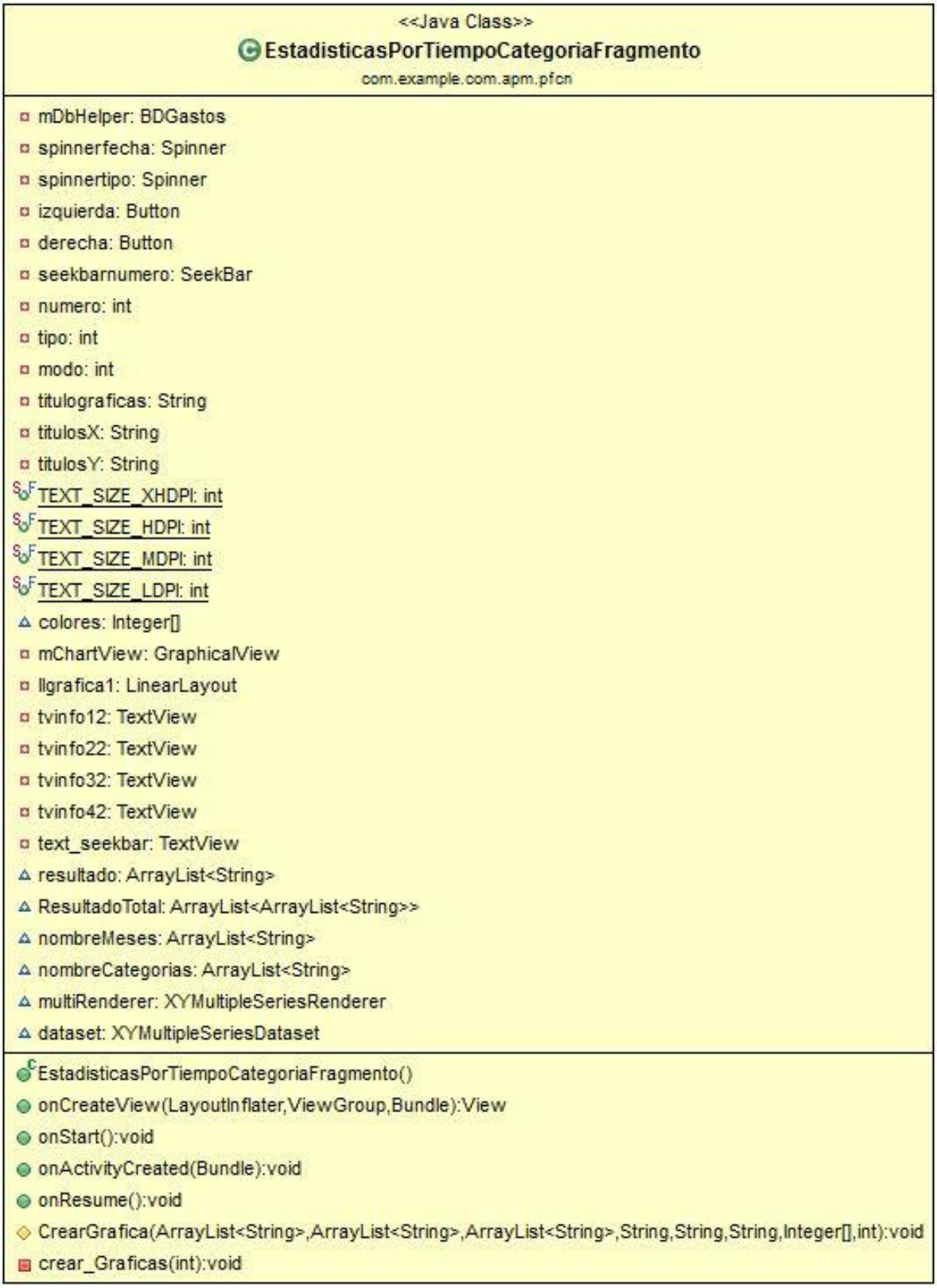
Es un fragmento más complicado por el alto número de componentes que lo forman.

En el siguiente código podemos ver como creamos una gráfica dinámicamente:

```
protected void CrearGrafica(ArrayList<String> res, ArrayList<String> nombres, String
        titulografica, String tituloX,
                    String tituloY) {
        XYSeries serie = new XYSeries("Gastos");
        XYSeriesRenderer renderer;
        dataset = new XYMultipleSeriesDataset();
        int listSize = res.size();
        int listSize2 = nombres.size();
```

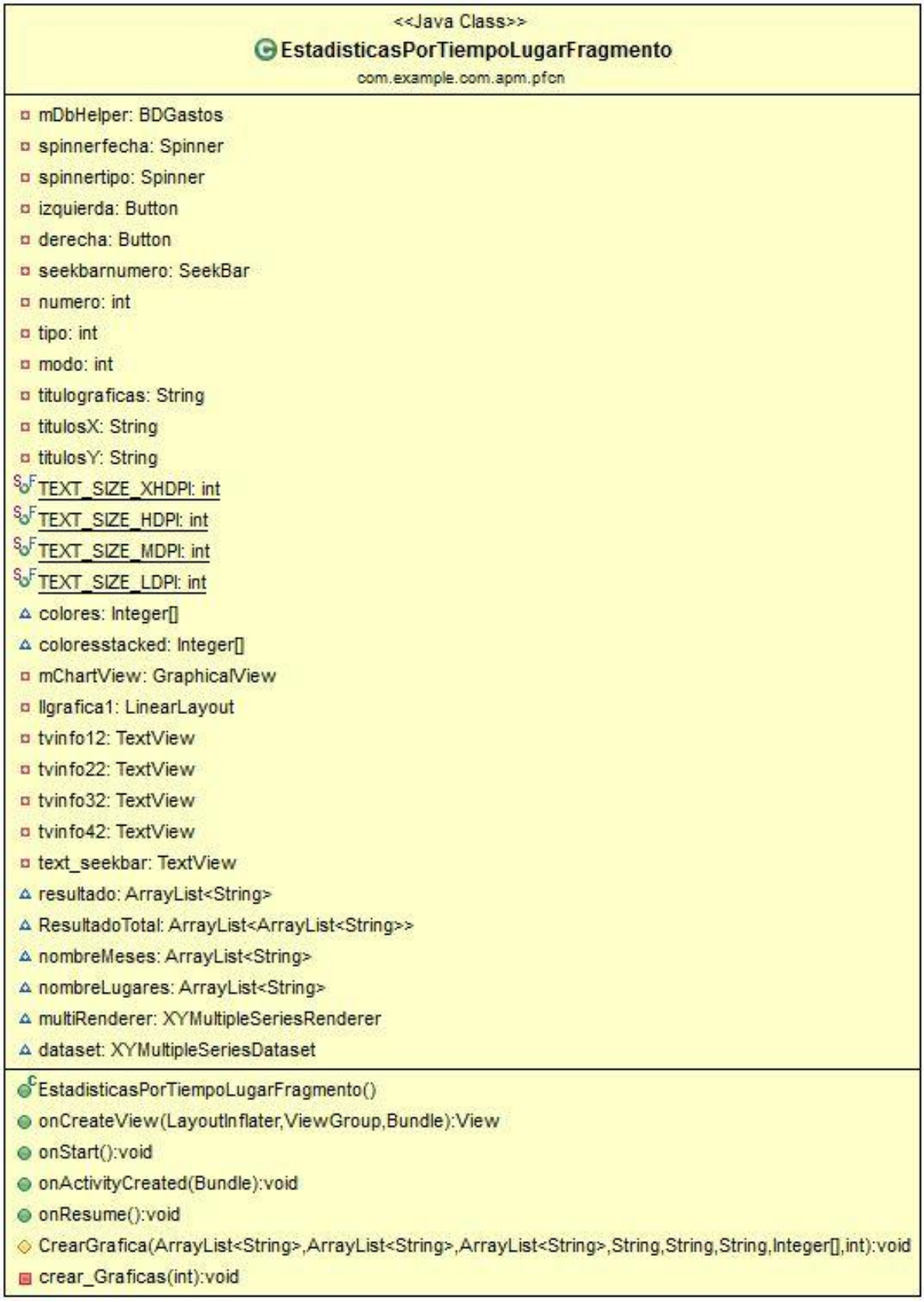
```
        renderer = new XYSeriesRenderer();
        renderer.setColor(Color.rgb(130, 130, 230));
        renderer.setFillPoints(true);
        renderer.setLineWidth(2);
        renderer.setDisplayChartValues(true);

        for (int i = 0; i < listSize; i++) {
              serie.add(i, Double.valueOf(res.get(i)));
        }
        dataset.addSeries(serie);
        multiRenderer = new XYMultipleSeriesRenderer();
        multiRenderer.setXLabels(0);
        multiRenderer.setChartTitle(titulografica);
        multiRenderer.setXTitle(tituloX);
        multiRenderer.setYTitle(tituloY);
        multiRenderer.setZoomButtonsVisible(true);
        multiRenderer.setZoomRate(1.1f);
        multiRenderer.setBarSpacing(0.5f);
        for (int i = 0; i < listSize2; i++) {
              multiRenderer.addXTextLabel(i, nombres.get(i));
        }
        multiRenderer.addSeriesRenderer(renderer);
        // añadimos las gráficas a su respectivo linear layout
        mChartView = ChartFactory.getBarChartView(getActivity(), dataset,
        multiRenderer, Type.DEFAULT);
        llgrafica1.removeAllViews();
        llgrafica1.addView(mChartView);
}
```

- **ESTADÍSTICAS POR CATEGORÍA TIEMPO FRAGMENTO:**



Este fragmento es el encargado de mostrar en forma de barras información sobre los gastos por categoría y por tiempo.

- **ESTADÍSTICAS POR LUGAR TIEMPO FRAGMENTO:**



Este fragmento es equivalente a "Estadísticas por categoría tiempo fragmento", realiza las mismas funciones pero respecto a los lugares de la base de datos.

5.3.4.6 FRAGMENTO FUNCIONALIDAD OCR

- **FRAGMENTO OCR:**

El fragmento OCR es el que contiene la funcionalidad para usar reconocimiento óptico en la aplicación.

Tiene diversas funciones para tomar fotos, obtener el OCR, mostrar la foto, y llamar al fragmento crear nuevo gasto.

El proceso que hay que realizar en este fragmento consiste en el siguiente:

- Tomar una foto.
- Seleccionar una parte de la foto para usarla como nombre (opcional).
- Seleccionar una parte de la foto para usarla como valor.
- Seleccionar una parte de la foto para usarla como descripción.
- Para cada una de estas partes de las fotos se obtiene el reconocimiento óptico de la foto, la información se guarda en un String.
- Después se procede a enviar esta información obtenida al fragmento nuevo gasto, en el cual se mostraran los datos obtenidos, y si el usuario considera adecuado guardarlos o modificarlos.

Ejemplos de código:

Aquí podemos ver como se realiza el proceso de tomar una foto:

```
public void takePhoto() {
        Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
        Calendar ahora = Calendar.getInstance();
        String nombreImagen = "Gasto " + Long.toString(ahora.getTimeInMillis()) +
        ".jpg";
        File photo = new File(Environment.getExternalStorageDirectory(), nombreImagen);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photo));
        imageUri = Uri.fromFile(photo);
        getActivity().startActivityForResult(intent, 100);
}
```

Esta es la función que se usa para obtener en un String el texto obtenido de la imagen mediante OCR:

La imagen se tiene que rotar, y colocar en el formato adecuado, en este caso “ARGB_8888”.

Rotar una imagen en Android puede causar problemas de memoria, en muchos casos tenemos que rotar imágenes de más de 8 megapíxeles, la memoria que puede usar una sola aplicación está limitada para no saturar todo el sistema.

```
protected String ObtenerOcr(Uri imageUri2, String nombreidioma) {
        if(idioma!=null){
             TessBaseAPI baseApi = new TessBaseAPI();
             Bitmap bitmap2 = null;
             BitmapFactory.Options options = new BitmapFactory.Options();
             options.inSampleSize = 2;
             bitmap2 = BitmapFactory.decodeFile(imageUri2.getPath(),options);
             ExifInterface exif2 = null;

             try {
                   exif2 = new ExifInterface(imageUri2.getPath());
             } catch (IOException e) {
                   e.printStackTrace();
             }
             int exifOrientation =
        exif2.getAttributeInt(ExifInterface.TAG_ORIENTATION,
        ExifInterface.ORIENTATION_NORMAL);

             int rotate = 0;
             switch (exifOrientation) {
             case ExifInterface.ORIENTATION_ROTATE_90:
                   rotate = 90;
                   break;
             case ExifInterface.ORIENTATION_ROTATE_180:
                   rotate = 180;
                   break;
             case ExifInterface.ORIENTATION_ROTATE_270:
                   rotate = 270;
                   break;
             }

             if (rotate != 0) {
                   int w = bitmap2.getWidth();
                   int h = bitmap2.getHeight();
```

```
                // Setting pre rotate
                Matrix mtx = new Matrix();
                mtx.preRotate(rotate);

                // Rotating Bitmap & convert to ARGB_8888, required by tess
                bitmap2 = Bitmap.createBitmap(bitmap2, 0, 0, w, h, mtx,
        false);
            }
            bitmap2 = bitmap2.copy(Bitmap.Config.ARGB_8888, true);
            baseApi.init("/mnt/sdcard/tesseract-ocr", nombreidioma);
            baseApi.setImage(bitmap2);
            String recognizedText = baseApi.getUTF8Text();
            Log.i("AAAAA", recognizedText);
            baseApi.end();
            bitmap2.recycle();
            return recognizedText;
            }else{
                AlertDialog.Builder builder = new
        AlertDialog.Builder(getActivity());
                builder.setTitle("Idioma no instalado");
                builder.setMessage("Debe instalar un idioma desde gestion de
        idiomas");
                builder.setPositiveButton("OK", null);
                AlertDialog dialog = builder.show();
                return "";
            }

        }
```

5.3.4.7 ACTIVIDAD DE GESTIÓN DE PREFERENCIAS

- **PREFERENCES ACTIVITY:**

Es una clase muy simple, ya que la mayoría de la funcionalidad de esta clase se realiza en las diferentes clases.

5.3.4.8 CLASE SOPORTE

- **SOPORTE:**

En soporte se encuentran una gran cantidad de funciones que he ido desarrollando mientras he realizado el PFC, muchas de ellas son tratamientos de strings o Lists, obtener fechas, o para solucionar problemas específicos de Android.

Ejemplos:

getIndex es una función creada para solucionar un problema que tiene Android cuando se usan spinner con un adaptador a una base de datos, el objeto Spinner de Android no permite buscar en qué posición esta un valor determinado ya que los valores son obtenidos de una base de datos.

```
public static int getIndex(Spinner spinner, String myString) {
        int index = 0;
        String String;
        Cursor cursor;
        for (int i = 0; i < spinner.getCount(); i++) {
```

```
                cursor = (Cursor) spinner.getItemAtPosition(i);
                String = cursor.getString(1);
                if (string.equals(myString)) {
                        index = i;
                }
        }
        return index;
}
```

Esta simple función **Obtenerfecha,** nos devuelve en milisegundos la hora actual.

```
public static long Obtenerfecha() {
                Calendar hoy = Calendar.getInstance();
                return hoy.getTimeInMillis();
        }
```

TruncateToString, es una función que elimina todos los decimales menos 2 de un valor doublé, y lo convierte a String:

```
public static String truncateToString(double value, int places) {
                if (places < 0) {
                        throw new IllegalArgumentException();
                }

                long factor = (long) Math.pow(10, places);
                value = value * factor;
                long tmp = (long) value;
                return new DecimalFormat("00.00").format((double) tmp / factor);
        }
```

5.3.5 INTERFAZ DE USUARIO

5.3.5.1 NAVEGACIÓN POR LA APLICACIÓN

Una de las principales decisiones en una aplicación de Android es como se realizara la navegación por la aplicación.

Existes diversas formas o estándares para implementar la navegación por las pantallas de la aplicación:

- **Una pantalla con un menú principal**: Este es el método usado por aplicaciones más antiguas, y no es recomendado por google ya que hace parecer la aplicación antigua, y poco usable.
- **Spinner de selección de pantallas:** Es un método usado en algunas aplicaciones, consiste en colocar un spinner en la barra de acción de la aplicación para que el usuario pueda elegir que pantalla quiere usar.
- **Fixed Tabs**: Este sistema se usa cuando la aplicación tiene pocas pantallas, al usuario se le muestra una pequeña barrar donde puede elegir que pantalla quiere usar. También puede cambiar de pantalla mediante un gesto a la izquierda o a la derecha
- **Scrollable tabs**: es igual que fixed tabs, pero es más dinámico, ya que en pantalla no tienen por qué aparecer todos los tabs disponibles.
- **Navigation drawer**: Es una barra lateral con un menú que aparece cuando el usuario hace un gesto de izquierda a derecha, o apretar el botón de la aplicación en la barra de acción. Es el nuevo sistema de navegación usado por las aplicaciones de Google.

Para la aplicación he decido usar Navigation drawer debido a la cantidad de pantallas que tiene la aplicación, y por qué es el sistema de navegación que recomienda Google en las aplicaciones de este tipo.

5.3.5.2 PANTALLAS DE LA APLICACIÓN

Para crear una pantalla de una aplicación de Android se debe definir de la siguiente forma:

- Se define la pantalla mediante un fichero XML, este fichero se debe crear en el directorio res de la aplicación, y dependiendo de la resolución se da un nombre al directorio. Por defecto todos los XML que definen las pantallas de una aplicación se guardan en el directorio “res\layout”, y dependiendo de la resolución e orientación se guardan en otras carpetas como “res\layout-land” o “res\layout-xhdpi”.
- El fichero XML, está formado por diferentes layouts (relative, linear), que pueden tener como hijos otros layouts o los diferentes widgets disponibles en Android (Button, Spinner).

Ejemplo de layout de la aplicación:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/linear1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <TextView
        style="@style/AppBaseTheme"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ellipsize="end"
        android:gravity="right"
        android:maxLength="16"
        android:singleLine="true"
        android:text="@string/nombre"
        android:textSize="15sp" />

    <TextView
        style="@style/AppBaseTheme"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ellipsize="end"
        android:gravity="right"
        android:maxLength="16"
        android:singleLine="true"
        android:text="@string/descripcion"
        android:textSize="15sp" />

    <TextView
        style="@style/AppBaseTheme"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:ellipsize="end"
        android:gravity="right"
        android:maxLength="12"
        android:singleLine="true"
        android:text="@string/color"
        android:textSize="15sp" />

</LinearLayout>
```

- **MENÚ DE NAVEGACIÓN:**

En esta pantalla se puede ver el detalle del menú de navegación, a este menú se puede acceder mediante el botón de la esquina superior izquierda de la aplicación, o mediante un gesto de izquierda a derecha desde el extremo izquierdo de la pantalla. Se puede acceder a este menú en cualquier momento.

En todo momento el usuario podrá crear un gasto nuevo mediante el botón "+" en la barra de acción.

- **PANTALLA RESUMEN:**

En esta pantalla se muestra el resumen de los últimos gastos, junto a diversas estadísticas de los gastos del último mes.

Además se comparan los gastos realizados con el presupuesto mensual definido por el usuario.

- **PANTALLA LISTADO GASTOS:**

En la pantalla de listado de gastos se nos muestran todos los gastos introducidos en la base de datos de la aplicación.
Desde esta pantalla se pueden filtrar por los distintos campos de un gasto y ordenarlos.

Mediante una barra de selección de rango se pueden filtrar los gastos por rango de fechas.

- **PANTALLA NUEVO GASTO:**

En esta pantalla se introducen los datos del nuevo gasto que queremos crear.

Mediante Spinners se puede seleccionar la categoría, y lugar del gasto.

La fecha se selecciona mostrando un popup con un widget para seleccionar la fecha.

También desde esta pantalla se puede acceder a las pantallas para crear una nueva categoría, y un nuevo lugar de gasto.

Si el usuario introduce valores incorrectos, se le mostrara un mensaje de error.

- **PANTALLA DETALLE GASTO:**

En esta pantalla se muestra la información de los gastos introducida anteriormente.

Se pueden modificar y borrar el gasto si el usuario lo desea, pidiendo una confirmación si desea borrar el gasto.

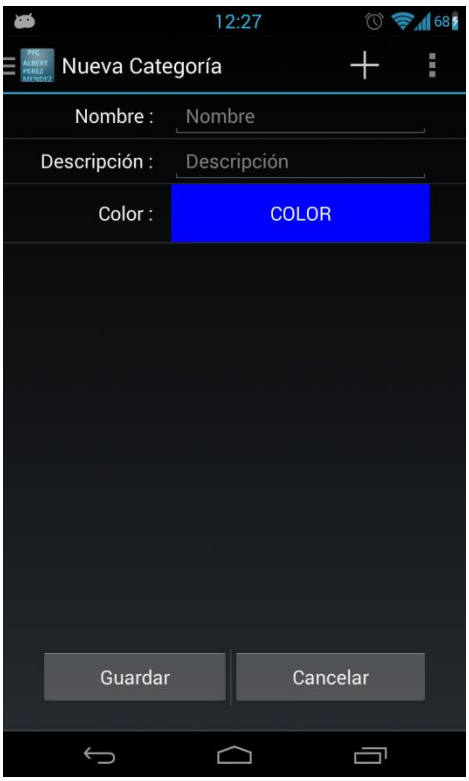
Si el usuario introduce valores incorrectos, se le mostrara un mensaje de error.

- **PANTALLA LISTADO CATEGORÍAS:**

En esta pantalla se nos muestran las diferentes categorías de gasto de la aplicación.

Desde esta pantalla el usuario puede acceder a la pantalla para crear una nueva categoría o a la pantalla para editar una categoría.

- **PANTALLA NUEVA CATEGORÍA:**

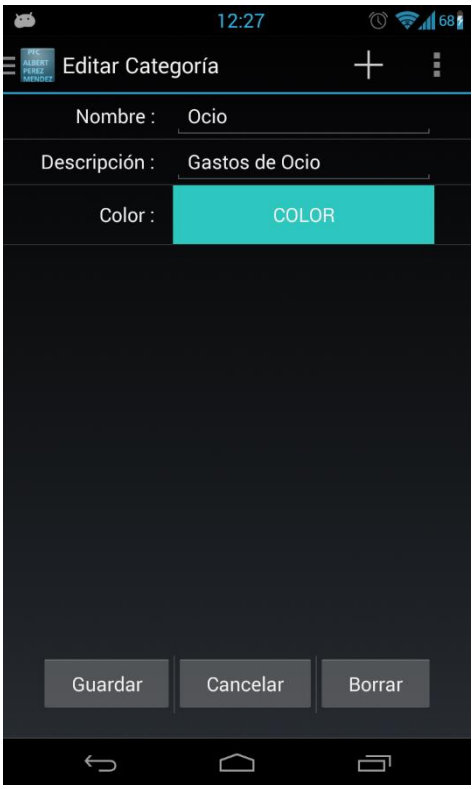


Desde esta pantalla el usuario puede introducir los datos de una nueva categoría.

Para seleccionar el color de la categoría se muestra al usuario un popup con un widget para seleccionar el color.

Si el usuario introduce valores incorrectos, se le mostrara un mensaje de error.

- **PANTALLA DETALLE CATEGORÍA:**



En esta pantalla el usuario de la aplicación puede modificar los valores de una categoría de gasto, o eliminarla de la base de datos.

Si el usuario introduce valores incorrectos, o intenta borrar una categoría en uso se le mostrara un mensaje de error.

- **PANTALLA LISTADO LUGARES:**

En esta pantalla al usuario se le muestran los diferentes lugares de gasto de la aplicación.

Desde esta pantalla el usuario puede acceder a la pantalla para crear un nuevo lugar o a la pantalla para editar un lugar.

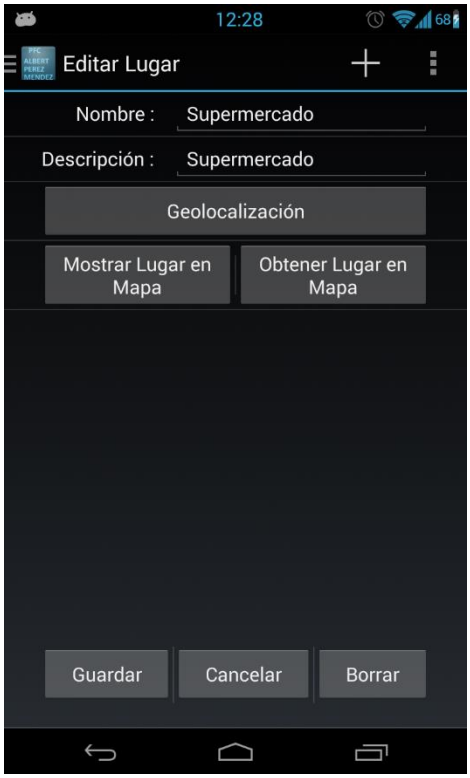
- **PANTALLA NUEVO LUGAR:**

Desde esta pantalla el usuario puede introducir los datos de un nuevo lugar.

El usuario puede obtener las coordenadas de la aplicación mediante el botón "Geo localización", o mediante los botones para mostrar un mapa.

Si el usuario introduce valores incorrectos, se le mostrara un mensaje de error.

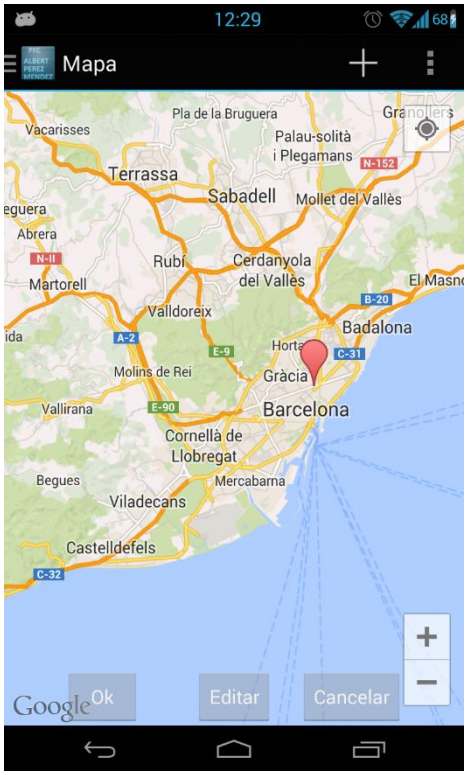
- **PANTALLA DETALLE LUGAR:**



En esta pantalla el usuario de la aplicación puede modificar los valores de un lugar de gasto, o eliminarlo de la base de datos

Si el usuario introduce valores incorrectos, o intenta borrar un lugar en uso se le mostrara un mensaje de error.

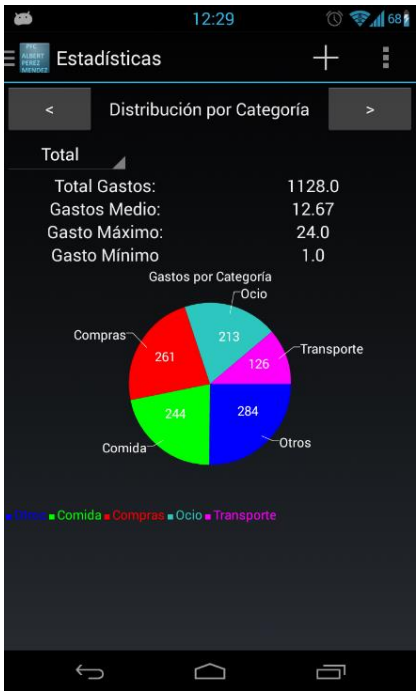
- **PANTALLA DE MAPA:**

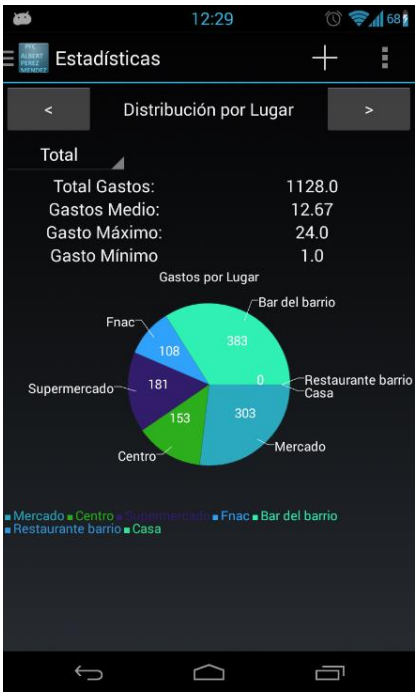


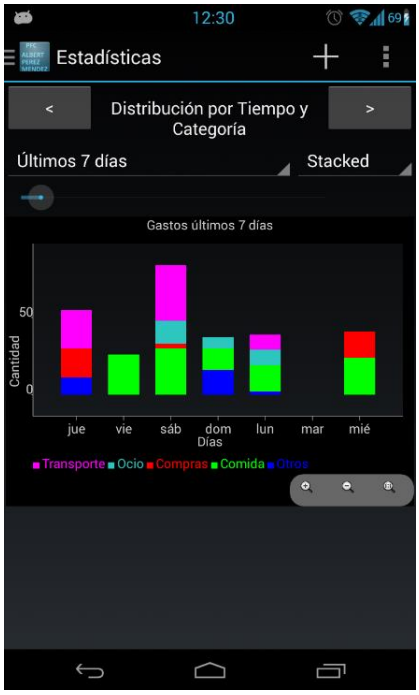
La pantalla de mapa nos permite mostrar u obtener la posición en el mapa del lugar de gasto.

Esta pantalla utiliza la Api de google Maps para mostrar la información y el mapa en pantalla.

- **PANTALLAS DE ESTADÍSTICAS:**







Diferentes pantallas con información estadística y graficas de los gastos introducidos en la aplicación.

La navegación por estas pantallas se realiza mediante los botones en la parte superior de la pantalla.

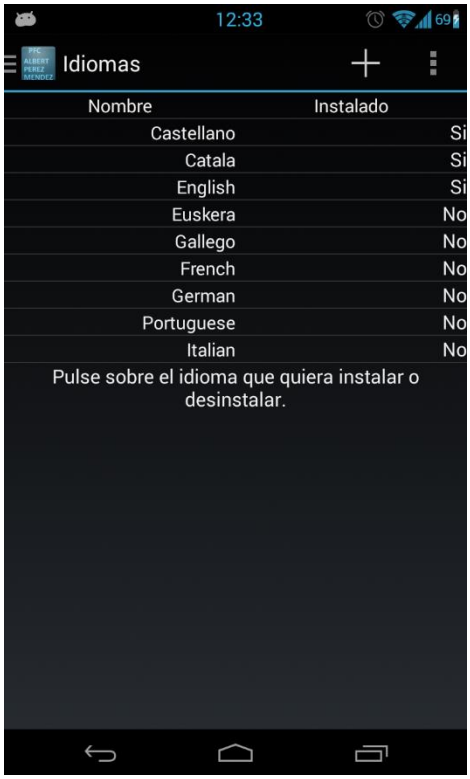
• **PANTALLA DE OCR:**

Desde esta pantalla se realizan las funciones de reconocimiento óptico de caracteres de la aplicación.

Mediante un Spinner se puede seleccionar el idioma.

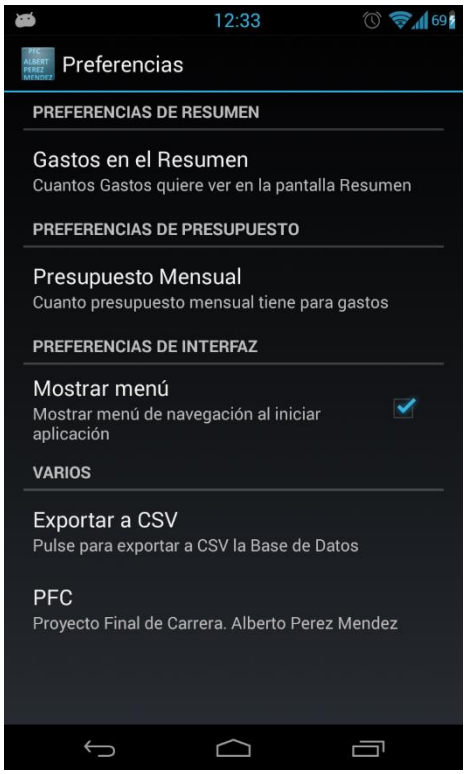
El usuario puede seleccionar una imagen de la galería o tomar una imagen con la cámara.

• **PANTALLA GESTIÓN DE IDIOMAS:**



Desde esta pantalla el usuario de la aplicación puede instalar o desinstalar los archivos de idioma para la funcionalidad OCR.

• **PANTALLA DE PREFERENCIAS:**



En la pantalla de preferencias el usuario de la aplicación puede modificar el funcionamiento de algunos detalles de la aplicación.

También desde esta pantalla puede acceder a la función para exportar la base de datos de la aplicación a CSV.

• **PANTALLAS EN ORIENTACIÓN HORIZONTAL:**

Debido a que algunas pantallas al colocar el móvil en posición horizontal no se mostraban correctamente, se han tenido que definir algunas de estas pantallas con diferente layout.

De forma automática la aplicación usa definiciones de pantalla diferentes dependiendo de la orientación delo móvil. Ejemplo:

5.3.5.2 LOCALIZACIÓN DE LA APLICACIÓN.

La aplicación esta traducida a 3 idiomas, que son castellano, catalán e inglés.

Para localizar una aplicación de *android* hay que realizar los siguientes pasos:

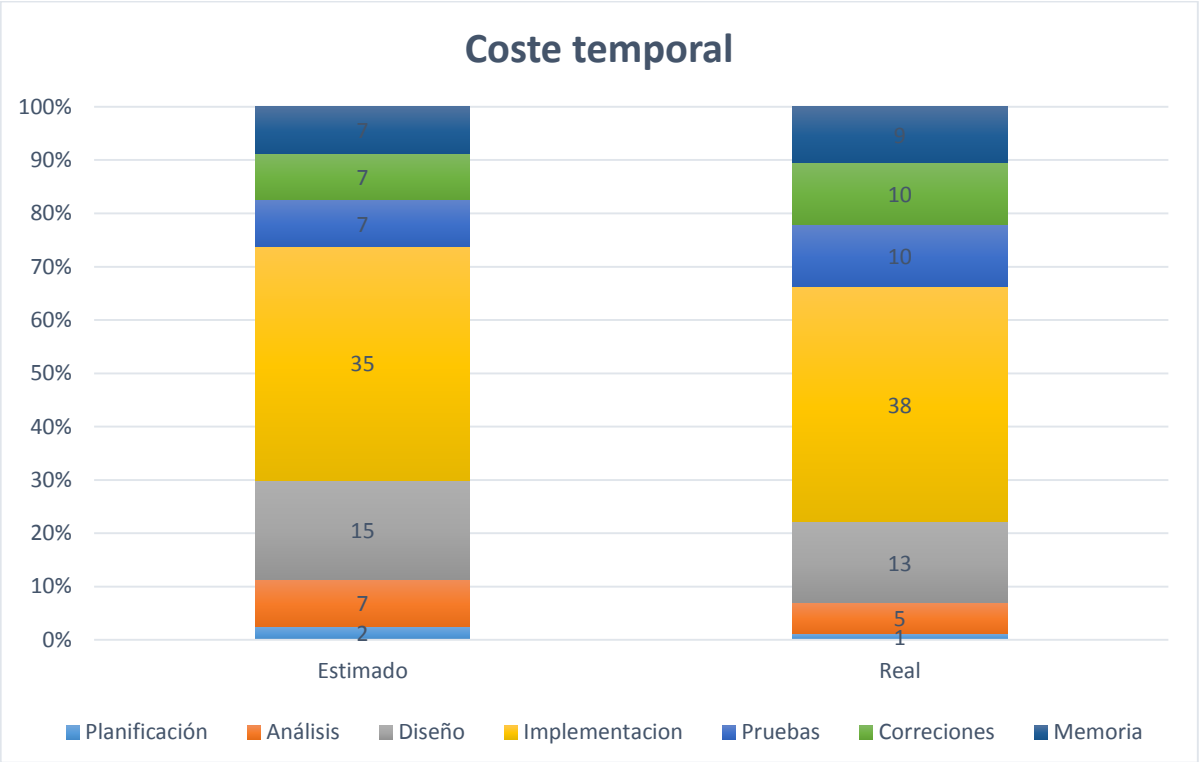
- Localizar todas las cadenas de texto usadas en los ficheros XML que definen las pantallas de la aplicación.
- Hay que crear un fichero en la carpeta res\values\ con el nombre string.xml, y asignas todas las cadenas de texto que hemos localizado en los ficheros XML a un string.
- Repetir estos 2 pasos, pero ahora localizamos todas las posibles cadenas de texto usadas en el código de la aplicación, como por ejemplo cuando creamos un widget de forma dinámica como puede ser un cuadro de dialogo.
- Una vez completado el fichero string.xml, se crea una nueva carpeta res\values para cada idioma que queremos traducir, por ejemplo en el caso del castellano se crear una carpeta res\values-es\.
- Se traducen los diferentes Strings de la aplicación, el sistema operativo Android dependiendo del idioma seleccionado del sistema operativo, usara el fichero string.xml adecuado.

6 PLANIFICACIÓN FINAL Y ANÁLISIS ECONÓMICO

6.1 COSTE TEMPORAL

La siguiente tabla nos muestra los distintos costes temporales divididos por recursos de cada una de las tareas, tanto de la planificación inicial que hemos mostrado anteriormente en la memoria, como de la planificación final o real del proyecto.

TAREA	**ANALISTA** (días estimados)	**PROGRAMADOR** (días estimados)	**ANALISTA** (días reales)	**PROGRAMADOR** (días reales)
Organización y planificación del proyecto	2		1	
Análisis de la aplicación	7		5	
Diseño de la aplicación	15		13	
Implementación		35		38
Test y pruebas		7		10
Correcciones		7		10
Realización de la memoria	7		9	
TOTAL	**31**	**49**	**28**	**58**
TOTAL DÍAS	**80**		**86**	
Meses	**3.63 Meses**		**3.9 Meses**	
Horas	**640 Horas**		**688 Horas**	



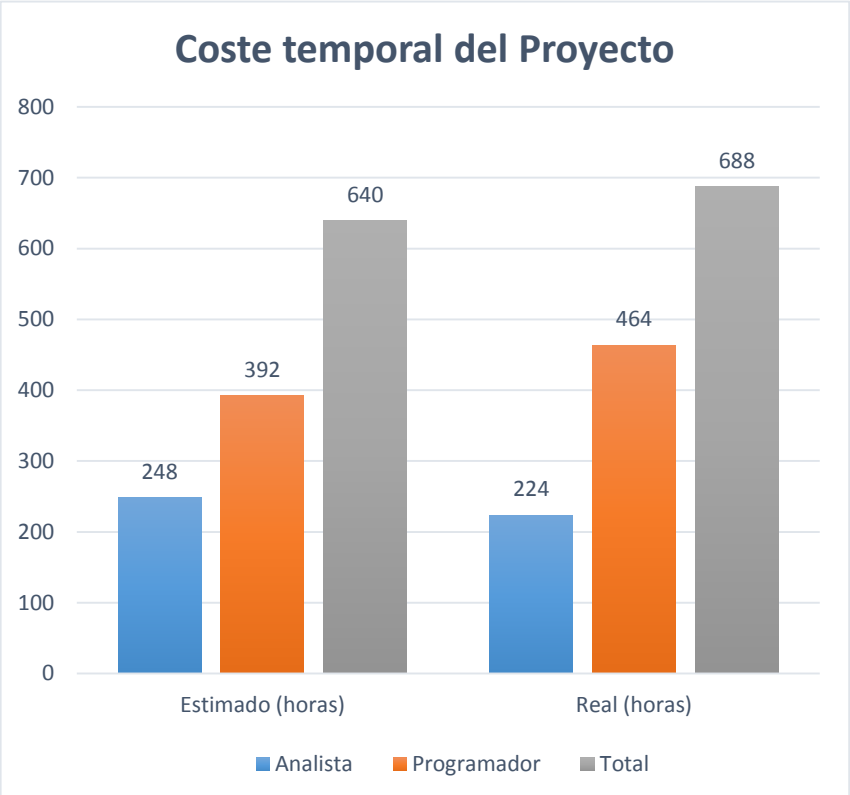
Como se puede ver en la tabla las principales diferencias temporales son:

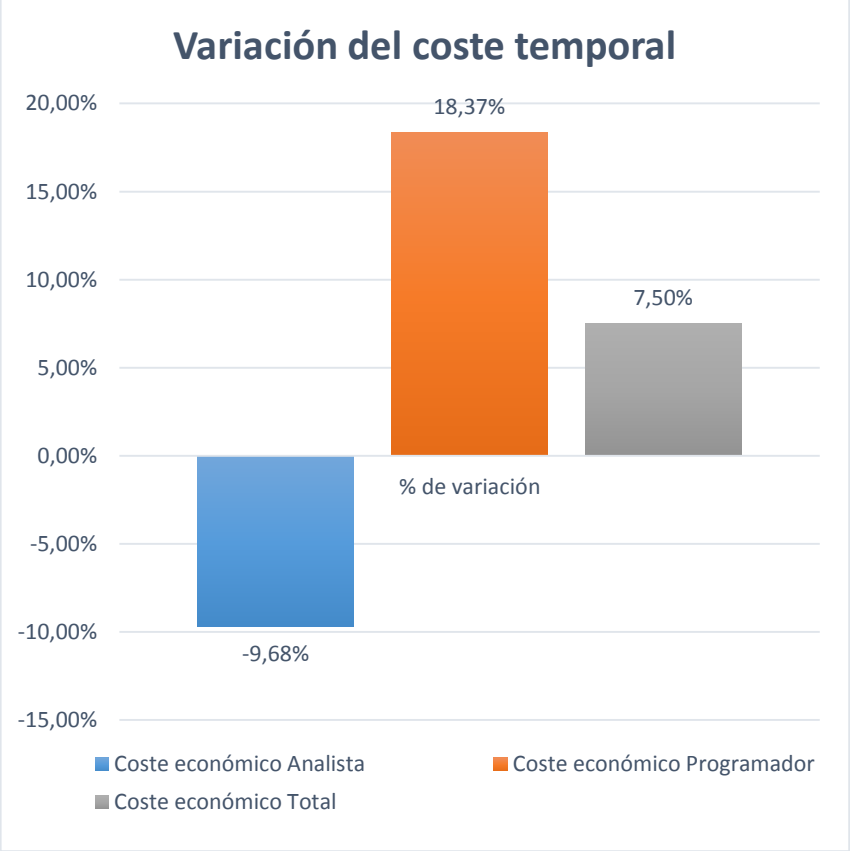
La realización de la organización, planificación, análisis y diseño han sido inferiores en la realidad, se había previsto un total de 31 días, y han resultado en la realidad 28 días, principalmente debido al menor coste temporal en el análisis y diseño de la aplicación. El único apartado que ha aumentado su coste temporal en esta parte ha sido la realización de la memoria.

Por otra parte, los costes temporales respecto a la implementación, test y corrección de pruebas han sido superiores a la prevista. El utilizar una plataforma como es Android que no se conocía antes de la realización del proyecto ha incrementado el tiempo necesario para implementar la aplicación. El incremento en este caso ha sido de 9 días.

El coste temporal total varía solo en 6 días, ya que el tiempo usado de más en la implementación y aprendizaje de tecnologías, se compensa con la reducción de tiempo usado en el análisis y diseño de la aplicación, en conclusión se puede resumir indicando que el coste temporal ha sido un 7,5% superior.

A todo este coste temporal se debería añadir el tiempo utilizado para aprender la tecnología utilizada, que ha sido aproximadamente de 2 semanas.


Coste temporal del Proyecto
800
700
600
500
400
300
200
100
0
248
392
640
224
464
688
Estimado (horas)
Real (horas)
Analista
Programador
Total


Variación del coste temporal
20,00%
15,00%
10,00%
5,00%
0,00%
-5,00%
-10,00%
-15,00%
-9,68%
18,37%
7,50%
% de variación
Coste económico Analista
Coste económico Programador
Coste económico Total

6.2 COSTE ECONÓMICO

Si consideramos el coste, para un empresario, por hora de un programador es de 35 euros, y el de un analista de 50 euros por hora, dentro de este coste por hora se incluyen todos los gastos asociados al realizar esta actividad y no solo el sueldo del trabajador, nos daría como resultado un coste de:

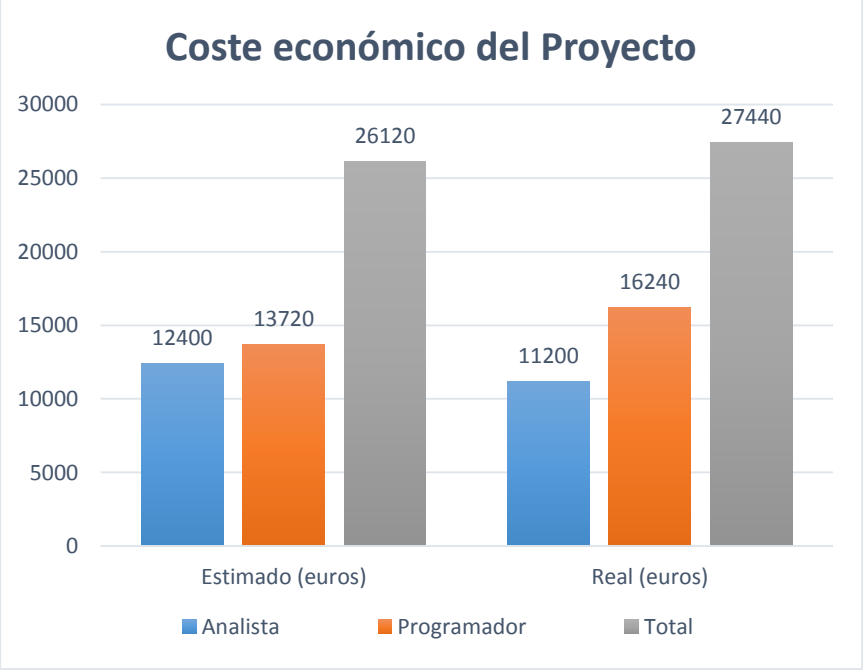
	ANALISTA (Estimado)	**PROGRAMADOR** (Estimado)	**ANALISTA** (Real)	**PROGRAMADOR** (Real)
Horas	248 Horas	392 Horas	224 Horas	464 Horas
Coste	12400 Euros	13720 Euros	11200 Euros	16240 Euros
COSTE TOTAL	**26120 Euros**		**27440 Euros**	

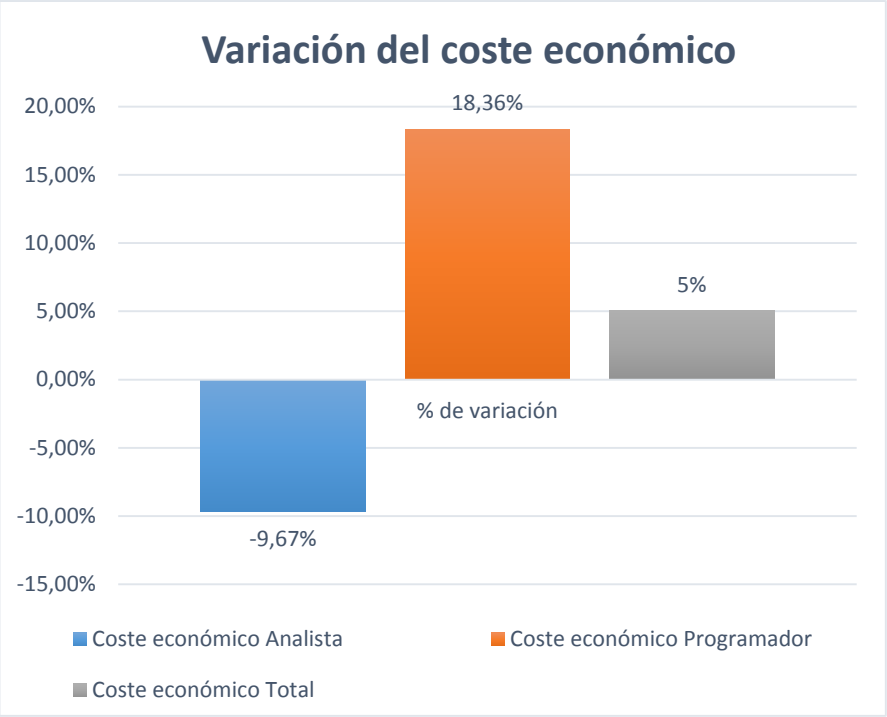
El coste del trabajo de analista se ha reducido en 1200 euros, ya que como hemos visto en el análisis de coste temporal, su trabajo se ha visto reducido en 24 horas en la planificación final.

Como contraposición a esta reducción de costes en el trabajo de analista, podemos ver que el coste del trabajo de programación se ha aumentado considerablemente, ya que el número de horas dedicado se incrementó en 72 horas. Esto da un coste total del trabajo de programador de 16240 euros, respecto a los 13720 presupuestados al inicio, en total es una diferencia de 2520 euros en el trabajo de programación respecto al presupuestado.

Sumando todos los costes económicos, el coste económico total del proyecto es de 27440 Euros. Como podemos ver en la tabla hay una diferencia de 1320 euros respecto al presupuesto inicial estimado que es de 26120 euros, en total el coste económico se ha visto incrementado un 5,05%.

Al igual que en el coste temporal, a todo este coste no se añade el coste económico resultante de tener que aprender una nueva tecnología como es Android, ya que es un coste que no se debería aplicar al usuario final de la aplicación.

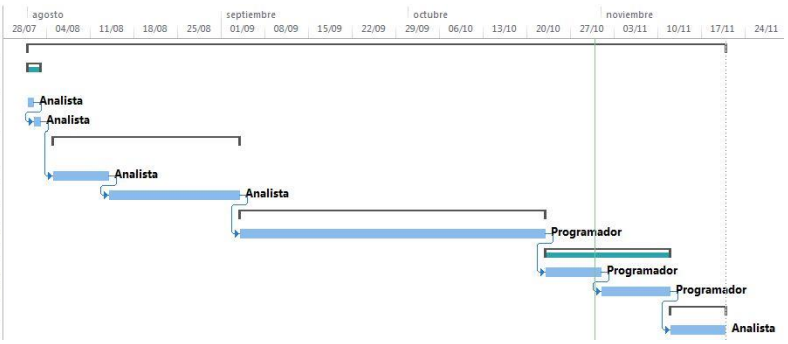



Variación del coste económico
20,00%
15,00%
10,00%
5,00%
0,00%
-5,00%
-10,00%
-15,00%
18,36%
5%
% de variación
-9,67%
Coste económico Analista
Coste económico Programador
Coste económico Total

6.3 DIAGRAMA DE GANTT

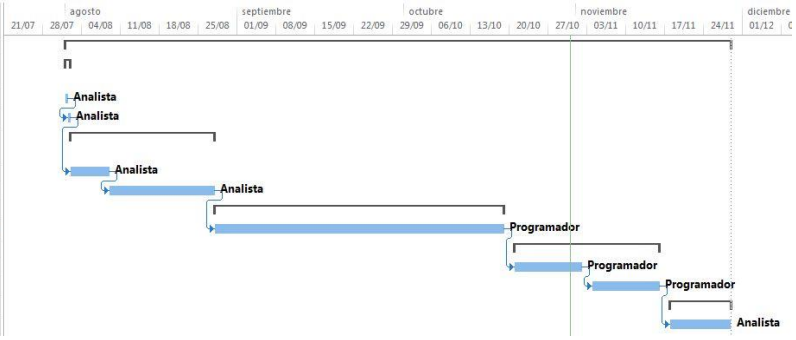
- **ESTIMADO**

	Nombre de tarea	Duración	Comienzo	Fin	Predecesc	Nombres de los recursos
1	**TOTAL**	**80 días**	**jue 01/08/13**	**mié 20/11/13**		
2	**Organización y planificación del proyecto**	**2 días**	**jue 01/08/13**	**vie 02/08/13**		
3	Planificación	1 día	jue 01/08/13	jue 01/08/13		Analista
4	Descripción aplicación	1 día	vie 02/08/13	vie 02/08/13	3	Analista
5	**Análisis y Diseño de la aplicación.**	**22 días**	**lun 05/08/13**	**mar 03/09/13**		
6	Analisis de la aplicación	7 días	lun 05/08/13	mar 13/08/13	4	Analista
7	Diseño de la aplicación	15 días	mié 14/08/13	mar 03/09/13	6	Analista
8	**Implementación**	**35 días**	**mié 04/09/13**	**mar 22/10/13**		
9	Implementación	35 días	mié 04/09/13	mar 22/10/13	7	Programador
10	**Tests y pruebas**	**14 días**	**mié 23/10/13**	**lun 11/11/13**		
11	Tests y pruebas	7 días	mié 23/10/13	jue 31/10/13	9	Programador
12	Correcciones	7 días	vie 01/11/13	lun 11/11/13	11	Programador
13	**Memoria**	**7 días**	**mar 12/11/13**	**mié 20/11/13**		
14	Realización de la memoria	7 días	mar 12/11/13	mié 20/11/13	12	Analista



- **REAL**

	Nombre de tarea	Duración	Comienzo	Fin	Predecesc	Nombres de los recursos
1	**TOTAL**	**86 días**	**jue 01/08/13**	**jue 28/11/13**		
2	**Organización y planificación del proyecto**	**1 día**	**jue 01/08/13**	**jue 01/08/13**		
3	Planificación	4 hrs	jue 01/08/13	jue 01/08/13		Analista
4	Descripción aplicación	4 hrs	jue 01/08/13	jue 01/08/13	3	Analista
5	**Análisis y Diseño de la aplicación.**	**18 días**	**vie 02/08/13**	**mar 27/08/13**		
6	Analisis de la aplicación	5 días	vie 02/08/13	jue 08/08/13	4	Analista
7	Diseño de la aplicación	13 días	vie 09/08/13	mar 27/08/13	6	Analista
8	**Implementación**	**38 días**	**mié 28/08/13**	**vie 18/10/13**		
9	Implementación	38 días	mié 28/08/13	vie 18/10/13	7	Programador
10	**Tests y pruebas**	**20 días**	**lun 21/10/13**	**vie 15/11/13**		
11	Tests y pruebas	10 días	lun 21/10/13	vie 01/11/13	9	Programador
12	Correcciones	10 días	lun 04/11/13	vie 15/11/13	11	Programador
13	**Memoria**	**9 días**	**lun 18/11/13**	**jue 28/11/13**		
14	Realización de la memoria	9 días	lun 18/11/13	jue 28/11/13	12	Analista



7 CONCLUSIONES

Al finalizar el proyecto, puedo decir sin ninguna duda que mi valoración del mismo es muy positiva. Ha sido una posibilidad de poner en uso herramientas y habilidades que he obtenido durante mis estudios, y además me ha permitido aprender nuevas tecnologías y plataformas.

Es muy satisfactorio para un estudiante ver como utilizando todos estos recursos que hemos aprendido podemos realizar tareas que antes de empezar a estudiar ni nos planteábamos.

Por supuesto no ha sido un proceso fácil la realización del proyecto final de carrera, desde la elección del proyecto pasando por el análisis y diseño, hasta la implementación, se nos plantean decisiones y dudas que han sido difíciles de resolver. Pero con la ayuda del trabajo realizado durante los estudios, dedicar muchas horas, y la ayuda de diferentes profesores, se han podido solventar de forma positiva todos estos problemas que se me han planteado durante el desarrollo del PFC.

Centrándonos en los objetivos iniciales del proyecto, puedo decir que se han cumplido todos los objetivos. Además durante la realización del proyecto se han ido añadiendo mejoras y otras funcionalidades que no estaban en los objetivos iniciales, como son la gestión de idiomas dentro de la aplicación, o mejoras en los filtros de gastos.

Estas problemáticas de las que hablo han sido las siguientes:

- Problemas con el análisis y diseño al utilizar la metodología UML, la falta de experiencia en este campo ha sido un gran problema, pero gracias, a recursos que puede obtener de mi tutor del proyecto Lluís Padró, y la ayuda de J.M. Merenciano, se han podido resolver.
- Falta de experiencia en la programación con Java, SQL y entorno Android. El pasar de un entorno controlado de programación como son las diferentes prácticas que hemos realizado durante los estudios, a un entorno nuevo para mí como es Android, ha resultado en una gran cantidad de dudas, y en tener que usar más tiempo del inicialmente planificado en aprender a desarrollar en este entorno.
- Utilizar librerías externas que requieren también un aprendizaje.
- Dificultad en cumplir con la planificación inicial, las diferentes problemáticas que he encontrado en el desarrollo del PFC han causado que se incremente el tiempo de realización de varias tareas del PFC.

Mejoras futuras de la aplicación:

Añadir la posibilidad de tener diferentes usuarios o cuentas dentro de la aplicación, solo si esta funcionalidad se puede añadir de forma simple para el usuario.

Añadir la posibilidad de asignar una divisa a los gastos, igualmente se necesita implementar de forma que sea fácil su uso, ya que los gastos del usuario en gran medida serán siempre en la misma divisa.

Mejoras en la introducción de información mediante OCR, como implementar dentro de la aplicación la función de tomar una foto sin usar una aplicación externa.

Posibilidad de obtener más información del gasto realizado desde la misma funcionalidad de OCR. Como por ejemplo, la fecha del gasto.

Añadir soporte para añadir gastos desde aplicaciones externas, como aplicaciones de gestión de cuentas bancarias por el móvil, o aplicaciones de pago por móvil.

Añadir otras estadísticas o gráficas a la aplicación que puedan mejorar la información que se le muestre al usuario.

8 BIBLIOGRAFÍA

- Wei-Meng Lee. Android 4, Desarrollo de aplicaciones. Ediciones ANAYA MULTIMEDIA, 1ª ed., 2012.
- Grant, Allen. Beginning Android 4. Apress. Edición: 1 (21 de diciembre de 2011)
- Larman, Craig. Applying UML and Patterns. Prentice-Hall, Edición: 3, 2006
- Friesen Jeff. Java para desarrollo Android. Ediciones ANAYA MULTIMEDIA, Edición: edición (23 de junio de 2011).
- Josep Maria Merenciano – Apunts INEP
- Android Developers Centre. http://developer.android.com/intl/es/index.html
- StackOverflow. http://stackoverflow.com/
- SQLite - http://www.sqlite.org/
- Tesseract Tools for Android. https://github.com/rmtheis/tess-two
- AChartEngine - http://www.achartengine.org/
- Range Seek Bar - https://code.google.com/p/range-seek-bar/
- Google Maps API - https://developers.google.com/maps/documentation/android/
- Android Dashboards - http://developer.android.com/intl/es/about/dashboards/index.html
- Java Developers Website - http://www.oracle.com/technetwork/java/index.html
- ABBYY ORC library - http://www.abbyyeu.com/
- JOCR Library - http://jocr.sourceforge.net/
- Android Plot - http://androidplot.com/
- Google Charts - https://developers.google.com/chart/
- ChartDroid - https://code.google.com/p/chartdroid/
- IDC, información sobre estado del mercado de sistemas operativos para dispositivos móviles - http://www.idc.com/getdoc.jsp?containerId=prUS24108913
- Expense Manager - https://play.google.com/store/apps/details?id=com.expensemanager&hl=es
- Expense Manager (Markus Hintersteiner) – https://play.google.com/store/apps/details?id=at.markushi.expensemanager
- Control de Gastos - https://play.google.com/store/apps/details?id=com.agudoApp.salaryApp&hl=es

9 ANEXOS

ANEXO 1: MANUAL

En este apartado se explica cómo utilizar la aplicación y como realizar las diversas funciones que permite la misma.

Navegar por la aplicación:

La navegación por la aplicación se realiza mediante el menú lateral, la barra de acción y el menú nativo de la aplicación.

- **Menú lateral:** Para acceder a este menú se puede simplemente apretar el botón mostrado en la parte superior izquierda de la pantalla, o haciendo un gesto en el extremo izquierdo de la pantalla hacia la derecha.

Desde este menú se puede acceder a la mayoría de las pantallas de la aplicación:

- Resumen
- Lista de gastos
- Lista de categorías
- Lista de lugares
- Estadísticas
- Funcionalidad de OCR
- Preferencias

- **Barra de acción:** Desde la barra de acción se puede en todo momento acceder a la opción de crear un gasto con el botón "+". Dependiendo de la pantalla en que este el usuario aparecerá otro icono, tanto para añadir una categoría, como para añadir un lugar.

También desde la barra de acción se puede acceder al menú nativo mediante el icono con 3 puntos, este menú también se puede acceder directamente si el dispositivo Android tiene un botón físico en el terminal para acceder al menú.

- **Menú nativo Android:** Desde este menú se puede acceder a las siguientes pantallas en todo momento:
  - Añadir nuevo gasto
  - Añadir nueva categoría
  - Añadir nuevo lugar
  - Añadir nuevo gasto mediante OCR
  - Preferencias
  - Exportar a CSV la base de datos

¿CÓMO AÑADIR UN GASTO?

Para acceder a la opción de añadir un gasto el usuario puede simplemente apretar el botón "+" de la barra de acción o seleccionar la opción en el menú nativo de Android.

Al usuario se le mostrara un formulario con los distintos campos que forman un gasto.

- Los campos de Nombre, Valor y Descripción se rellenan mediante un cuadro de texto.
- La fecha por defecto se mostrara la del día actual, pero el usuario puede pulsar el botón fecha que le mostrara un cuadro de dialogo para que puede seleccionar la fecha.
- La categoría y lugar se seleccionan mediante un Spinner o selector, si el usuario necesita crear una categoría nueva puede hacerlo apretando el botón "+" al lado de categoría y lugar.
- Si el usuario desea incluir una imagen, puede usar el botón "Tomar foto", que ejecutara la aplicación de cámara de fotos del sistema Android para realizar un foto.
- Si el usuario quiere guardar el gasto simplemente tiene que apretar el botón "Guardar", y si prefiere descartar los cambios puede usar el botón "Cancelar".

¿CÓMO AÑADIR UN GASTO MEDIANTE OCR?

Para acceder a la opción de añadir un gasto por OCR, el usuario puede seleccionar la opción OCR del menú lateral o seleccionar la opción en el menú nativo de Android.

El usuario deberá elegir mediante el selector de idioma el idioma en que esta el ticket o factura que desea usar.

Puede realizar una foto al ticket/factura, o elegir una foto de la galería de fotos si ha realizado la foto con anterioridad.

Después de realizar este paso se activaran los botones "nombre", "valor" y "descripción". Al pulsar uno de estos botones al usuario se le mostrará la foto, y podrá seleccionar en la imagen la parte del ticket que quiere usar como "nombre", "valor" y "descripción" respectivamente.

Una vez realizado esto, el usuario podrá seleccionar la opción crear gasto, que mostrara al usuario la pantalla añadir nuevo gasto con los valores que ha seleccionado mediante la funcionalidad OCR.

¿CÓMO VER/MODIFICAR/ELIMINAR UN GASTO?

La mecánica para modificar/eliminar un gasto es idéntica a añadir un gasto.

Se puede acceder a esta opción seleccionando un gasto en la pantalla resumen o en la pantalla de listado de gastos.

El usuario podrá simplemente consultar los detalles del gasto, o modificar el gasto mediante los diferentes widgets de la pantalla.

Si el usuario desea guardar los cambios realizados debe seleccionar la opción "Guardar", si desea descartar los cambios o no modificar el gasto debe seleccionar la opción "Cancelar".

El usuario también podrá eliminar el gasto mediante la opción "Borrar", al usuario se le mostrara una pantalla de confirmación.

¿CÓMO CONSULTAR LOS GASTOS?

El usuario puede consultar los gastos desde la opción "Gastos" del menú lateral.

En esta pantalla usando los Spinners o selectores arriba de la pantalla puede filtrar, y ordenar por los distintos campos disponibles los gastos.

También el usuario puede mediante la barra de selección de rango, elegir mostrar solo los gastos en un rango de fechas determinado por el usuario.

El usuario puede modificar o consultar los detalles de un gasto simplemente seleccionando un gasto, esto abrirá la pantalla de modificar gasto.

Debajo de la pantalla se puede ver el valor total de los gastos mostrados en ese momento en la pantalla.

¿CÓMO CONSULTAR ESTADÍSTICAS Y GRAFICAS?

El usuario puede acceder a esta funcionalidad mediante el menú lateral escogiendo la opción "Estadísticas". El usuario podrá navegar por las diferentes pantallas de estadísticas mediante los botones "<" y ">" de la parte superior de la pantalla.

En las pantallas el usuario podrá cambiar mediante un Spinner o seleccionador el rango de fechas de las estadísticas.

¿CÓMO EXPORTAR LA BASE DE DATOS A CSV?

Para exportar la base de datos a CSV el usuario puede o seleccionar la opción desde la pantalla de preferencias, o mediante el menú nativo de Android. El archivo CSV se creara en la raíz de la memoria del dispositivo.

¿CÓMO ACCEDER A LAS PREFERENCIAS DE LA APLICACIÓN?

Las preferencias de la aplicación se pueden acceder desde el menú lateral "Preferencias" o mediante la opción preferencias del menú nativo de Android.

¿CÓMO AÑADIR UNA CATEGORÍA?

Para acceder a la opción de añadir una categoría el usuario puede simplemente ir a la pantalla de listar categorías del menú lateral, y apretar el botón designado para tal uso en la barra de acción o seleccionar la opción en el menú nativo de Android.

Al usuario se le mostrara un formulario con los distintos campos que forman una categoría.

- Los campos de Nombre y Descripción se rellenan mediante un cuadro de texto.
- El color de la categoría se puede seleccionar mediante el botón color, que mostrara un cuadro de dialogo donde el usuario podrá seleccionar el color.
- Si el usuario quiere guardar la categoría simplemente tiene que apretar el botón "Guardar", y si prefiere descartar los cambios puede usar el botón "Cancelar".

¿CÓMO VER/MODIFICAR/ELIMINAR UNA CATEGORÍA?

Modificar/eliminar una categoría, se realiza desde una pantalla equivalente a crear una categoría.

Se puede acceder a esta opción seleccionando una categoría en la pantalla de listado de categorías.

El usuario podrá simplemente consultar los detalles de la categoría, o modificar el gasto mediante los diferentes widgets de la pantalla.

Si el usuario desea guardar los cambios realizados debe seleccionar la opción "Guardar", si desea descartar los cambios o no modificar el gasto debe seleccionar la opción "Cancelar".

El usuario también podrá eliminar la categoría mediante la opción "Borrar", al usuario se le mostrara una pantalla de confirmación. Si la categoría está en uso se le mostrara un mensaje de error al usuario.

¿CÓMO CONSULTAR LAS CATEGORÍAS?

El usuario puede consultar las categorías desde la opción "Categoría" del menú lateral.

En esta pantalla usando los Spinners o selectores arriba de la pantalla puede filtrar, y ordenar por los distintos campos disponibles las categorías.

El usuario puede modificar o consultar los detalles de una categoría simplemente seleccionando una categoría, esto abrirá la pantalla de modificar categoría.

¿CÓMO AÑADIR UN LUGAR?

Para acceder a la opción de añadir un lugar el usuario puede simplemente ir a la pantalla de listar lugares del menú lateral, y apretar el botón designado para tal uso en la barra de acción o seleccionar la opción en el menú nativo de Android.

Al usuario se le mostrara un formulario con los distintos campos que forman un lugar.

- Los campos de Nombre y Descripción se rellenan mediante un cuadro de texto.
- Si el usuario desea obtener la posición actual, deberá apretar el botón geo localización.
- Si desea obtener la posición actual desde el mapa, deberá usar el botón "Obtener lugar en mapa", se le mostrara un mapa al usuario y podrá seleccionar el lugar, y seleccionar "ok" para confirmar, o "cancelar" para descartar los cambios.
- Si el usuario quiere guardar el lugar tiene que apretar el botón "Guardar", y si prefiere descartar los cambios puede usar el botón "Cancelar".

¿CÓMO VER/MODIFICAR/ELIMINAR UN LUGAR?

Modificar/eliminar un lugar, se realiza desde una pantalla equivalente a crear una lugar.

Se puede acceder a esta opción seleccionando un lugar en la pantalla de listado de lugares.

El usuario podrá simplemente consultar los detalles del lugar, o modificar el gasto mediante los diferentes widgets de la pantalla, también podrá mostrar un mapa con el lugar del gasto.

Si el usuario desea guardar los cambios realizados debe seleccionar la opción “Guardar”, si desea descartar los cambios o no modificar el gasto debe seleccionar la opción “Cancelar”.

El usuario también podrá eliminar el lugar mediante la opción “Borrar”, al usuario se le mostrara una pantalla de confirmación. Si el lugar está en uso se le mostrara un mensaje de error al usuario.

¿CÓMO CONSULTAR LOS LUGARES?

El usuario puede consultar los gastos desde la opción “Lugares” del menú lateral.

En esta pantalla usando los Spinners o selectores arriba de la pantalla puede filtrar, y ordenar por los distintos campos disponibles los lugares.

El usuario puede modificar o consultar los detalles de un lugar simplemente seleccionando un lugar, esto abrirá la pantalla de modificar lugar.

ANEXO 2: RECONOCIMIENTO ÓPTICO DE CARACTERES

La tecnología de reconocimiento óptico de caracteres u OCR (Optical Character Recognition), son una seria de técnicas destinadas a procesar un documento escrito o foto, para poder obtener información, y después ser utilizada en un sistema informático. Este sistema facilita la entrada de información en un sistema informático.

Esta tecnología engloba una serie de técnicas, basadas en estadísticas, en las formas de caracteres o símbolos, transformaciones y comparaciones. Estas técnicas se complementan unas a otras, y permiten distinguir de forma automática caracteres alfanuméricos. De forma más precisa, se puede decir que no se reconocen los caracteres, sino unas formas o símbolos.

Esta serie de técnicas no tienen una fiabilidad del 100%, y dependiendo del tamaño del conjunto de símbolos también se ve reducida la efectividad de estas técnicas.

1 ETAPAS DE OCR

Un sistema de OCR está formado de las siguientes etapas:

- **Preproceso:** Adecuación de la imagen.
- **Segmentación:** Se selecciona la zona de interés de la imagen.
- **Extracción de características:** Representamos la imagen de forma digital.
- **Reconocimiento:** Se distinguen los diferentes caracteres de la imagen.

1.1 PREPROCESO

Durante este proceso se tiene que adecuar la imagen para las siguientes etapas. El objetivo es eliminar de la imagen ruidos o imperfecciones que no sean parte de un carácter, y normalizar el tamaño de estos caracteres.

Existen los siguientes algoritmos para realizar este proceso:

- **Expansión o erosión:** se utiliza para eliminar pixeles de tamaño muy reducido.
- **Etiquetado:** Se divide la imagen en regiones de componentes conectados.
- **Umbralizar histogramas:** Se eliminan o seleccionan los objetos más luminosos y oscuros de la imagen.

1.2 SEGMENTACIÓN

Esta etapa se realiza para descomponer el texto en diferentes entidades lógicas. Con esto obtenemos las diferentes componentes conexas de la imagen, que luego en las siguientes etapas se tendrán que distinguir en los diferentes caracteres.

Se han de detectar los diferentes, renglones o párrafos, las palabras que forman estos renglones, y las letras que forman las palabras.

Para detectar estos renglones se realizan 2 pasos:

- Se ha de realizar un histograma o proyección horizontal, con esto se cuentan los elementos que forman cada una de las filas, se traspasan estos valores a una matriz unidimensional, donde habrá zonas con diferentes densidades. Cada una de estas zonas tendrá un valor, y cuando el valor sea no nulo habremos obtenido hipotéticamente un renglón de texto.
- El segundo paso es distinguir en esta matriz unidimensional el inicio y final de esto renglones, buscando las líneas nulas antes y después del posible renglón de texto. Se realiza este proceso para toda la matriz, para separas los distintos renglones de texto.

Después de este proceso, se han de aislar los caracteres. Se realiza en este caso un proyección vertical dentro de cada renglón de texto obtenido, así se detectan los posibles caracteres. Se asigna 1 a las líneas con pixeles negros, y 0 a los blancos. Esta proyección da un resultado nulo en las líneas sin pixeles, lo que representa una separación de caracteres, y da un resultado no nulo cuando existe un carácter.

1.3 EXTRACCIÓN DE CARACTERÍSTICAS

Una vez realizada la segmentación, hemos obtenido una imagen normalizada donde tenemos la información que queremos reconocer. Esta imagen está representada en una matriz bidimensional, el tamaño de esta matriz es muy grande, y tiene demasiadas características no necesarias para reconocer los diferentes caracteres, intentar reconocer directamente los diferentes caracteres de esta matriz sería un gasto excesivo de recursos computacionales, y no se obtendrían un resultado satisfactorio.

Se han de aplicar una serie de técnicas para reducir el volumen de información, y extraer las diferentes características, estas características deben tener las siguientes cualidades:

- **Discriminación:** Debe diferenciar una clase con otra.
- **Valor:** Debe tener el mismo valor en las mismas clases.
- **Independiente:** Estas características no deben estar relacionadas unas con otras.
- **Numero:** El número de características debe ser limitado, para facilitar y aumentar la rapidez de la clasificación.

Se utilizan los siguientes métodos para extraer características:

- **PCA o principal component analysis:** Se realiza una transformación lineal de la representación original a una nueva representación donde las diferentes muestras están mejor separadas. Esta nueva representación consigue reducir el espacio utilizado, sin perjudicar las posibilidades de distinción de la muestra.
- **LDA o linear discriminant analysis:** LDA está basada en PCA, pero la técnica LDA además modela en esta nueva representación la diferencia de clases de datos.
- **NDA o non-linear Discriminan analysis:** Se obtiene la misma información que LDA pero sin representar los datos en combinaciones lineales.
- **ICA o independent Component Analysis:** ICA separa las señales en subcomponentes aditivas aplicando independencia estadística.

1.4 RECONOCIMIENTO

Para realizar la etapa de reconocimiento de caracteres se utilizan diferentes técnicas para determinar los mismos:

- **Algoritmo K-NN (vecinos más próximos):** Es un método muy utilizado por su sencillez, y es utilizado en diversos sistemas de clasificación. De forma estadística se compara el carácter a clasificar con una serie de muestras, y se asigna al carácter la clase "K" más numerosa de las muestras creadas anteriormente.

Se necesita crear primeramente un conjunto de muestras de clases o diccionario para poder comparar nuestros caracteres. Esta fase se llama fase de entrenamiento.

Estas muestras que pueden llegar a tener un número muy elevado de objetos, se debe guardar en estructuras complejas de datos (voronoy polygons, k-d-trees o r-trees).

- **Arboles de decisión:** Los arboles de decisión es otra técnica usada para OCR.

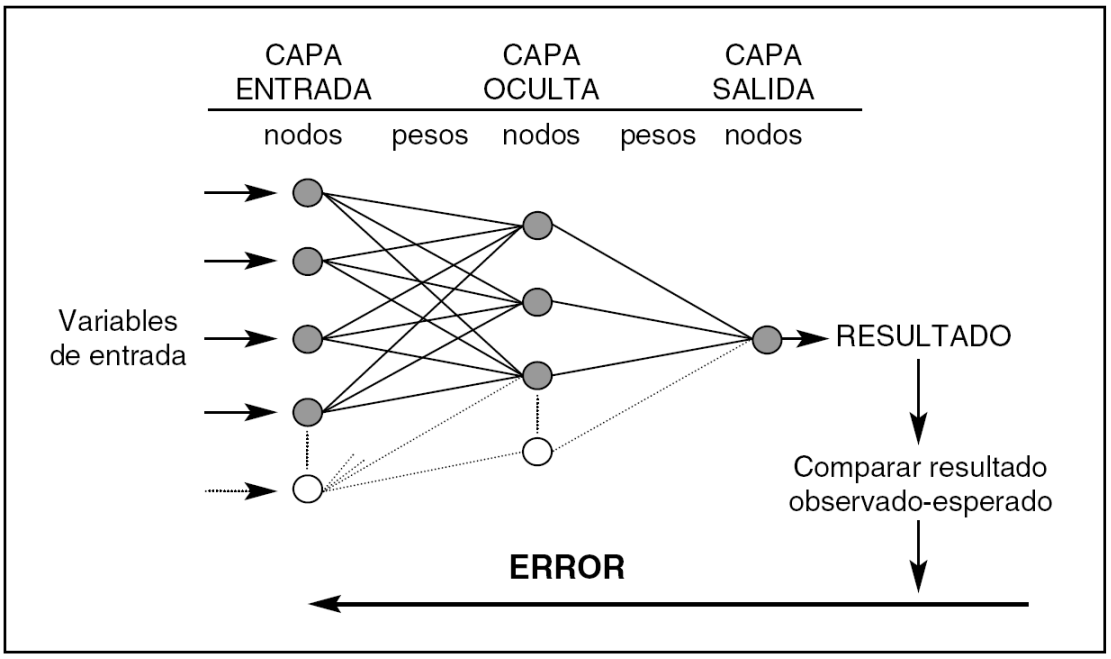
Cada atributo que deseamos evaluar es un nodo del árbol, y los resultados obtenidos son las hojas del árbol. Una vez construida la estructura del árbol, esta evaluación de caracteres se puede implementar mediante una arquitectura if, then y else. Esta estructura da como resultado que el coste computacional sea mucho menor al del algoritmo K-NN.

- **Redes neuronales:** Las redes neuronales son estructuras que intentan imitar la arquitectura del cerebro. Estas estructuras están formadas por neuronas, estas reciben una entrada, evalúan su peso y realizan una salida que dependen de la sumas de las salidas de la etapa anterior. Una vez entrenadas estas redes, se pueden utilizar para reconocer los caracteres de una imagen.

Las redes neuronales nos ayudan a representar funciones que son difíciles de representar en una forma algebraica.

Topologías de una red neuronal:

- **Feed forward:** Las salidas alimentan las entradas de la etapa siguiente, no existen ciclos o realimentaciones.
- **Feed back:** Si se permiten ciclos o realimentaciones.
- **Lateral:** Se permite además la comunicación vertical entre neuronas de la misma etapa.


CAPA ENTRADA
CAPA OCULTA
CAPA SALIDA
nodos
pesos
nodos
pesos
nodos
Variables de entrada
RESULTADO
Comparar resultado observado-esperado
ERROR

ANEXO 3: Distribuir una aplicación de Android.

La forma de distribuir una aplicación de Android, para que cualquier usuario del sistema operativo Android pueda acceder a ella, es publicar la aplicación en una o varias de las tiendas de aplicaciones disponibles. La tienda oficial, y mantenida por Google es la "Play store", pero también existen varias alternativas ofrecidas desde fabricantes de móviles a comunidades de usuarios.

1 PLAY STORE

Para que un desarrollador pueda publicar una aplicación en la Play Store, primero de todo tienes que darse de alta como desarrollador de aplicaciones para Android en la página oficial de Google para desarrolladores. Para darse de alta, el desarrollador debe aceptar una serie de condiciones/normas, y hacer una pago único de 25$ dólares. La cuenta de desarrollador se vinculara a una cuenta de google, después pasaremos a rellenar formulario con nuestra información, con información de contacto para que los usuarios de las aplicaciones puedan obtener ayuda.

La aplicación que queramos subir debe cumplir unos requisitos marcados por Google:

- **Tamaño del fichero APK de la aplicación**: la aplicación que queremos distribuir no debe superar los 50 megabytes, en caso de superar este tamaño se debe dividir la aplicación.
- **Capturas de pantalla:** La Play Store requiere como mínimo que se incluyan 2 capturas de pantalla de la aplicación en funcionamiento, esta se usaran en la página dedicada a la aplicación en la Play Store.
- **Icono de la aplicación**: Se debe incluir un icono identificativo de la aplicación, además este debe ser con contraste de alta resolución
- **Idioma:** Se debe indicar el idioma predeterminado de la aplicación, por defecto es el inglés, pero el desarrollador puede elegir publicarla por defecto en otro idioma.
- **Nombre de la Aplicación:** Se debe elegir un nombre por el que será conocida la aplicación, este idioma puede ser distinto según el idioma.
- **Descripción:** El desarrollador debe incluir una descripción de la aplicación, como máximo esta descripción debe ser de 4000 caracteres.
- **Países de distribución**: En este apartado se indicara si la aplicación en que países estará disponible la aplicación.

Si deseamos vender una aplicación en la Play Store habrá que tener unas consideraciones:

El 70% del dinero obtenido de las ventas de la aplicación será para el desarrollador, y el resto es para Google. Esta cifra es igual a la de otras tiendas de aplicaciones como la IOS store de Apple, o Windows store de Microsoft.

El IVA en las aplicaciones móviles será calculado mediante el País de residencia del usuario que compre la aplicación.

- Si el comprador es de España, el IVA que se cobrara es del 21%
- Si el comprador es de Europa, y no es empresario, se cobrara un 21%

- Si es comprador es de Europa, y es empresario con VAT o número de identificación fiscal europeo, el IVA será del 0%.
- Si el comprador es de fuera de la unión europea, el IVA cobrado será del 0%.

En el momento que como desarrollador cobremos algo por nuestras aplicaciones, deberemos darnos de alta como empresarios en la Agencia Tributaria. Esta alta se realiza mediante el modelo 037 (declaración censal de alta de actividades económicas).

Una vez hemos tramitado este proceso de alta, trimestralmente tendremos que presentar en la Agencia Tributaria el modelo 130 de pagos a cuenta en el IRPF, y otro modelo 303 que es la declaración trimestral de IVA. Para el modelo 130 de pagos a cuenta se calculara el total de ingresos menos el IVA, y menos los gastos deducibles, y se deberá ingresar trimestralmente el 20% del beneficio neto a Hacienda.

En el caso de la Seguridad Social, debido a un vacío legal, y a una sentencia del tribunal supremo del año 2007, se considera que si el desarrollador no supera el salario mínimo interprofesional, no deberá hacer el pago mensual de la cotización a la seguridad social. En caso de superarlo, sí que deberá realizar este pago que oscila entre 180 euros y 250 euros.

2 ALTERNATIVAS A GOOGLE PLAY STORE

Existen otra alternativas para distribuir nuestras aplicaciones a los usuarios, tanto alternativas ofrecidas por otras compañías, o creadas por comunidades de usuarios.

Las razones para publicar aplicaciones en otras tiendas de aplicaciones son entre otras las siguientes:

- No todos los países tienen acceso a la Play Store, esto limita que nuestra aplicación pueda llegar a algunos usuarios.
- No todos los dispositivos soportan la Play Store, o distribuyen la aplicación de la Play Store en su sistema operativo, hay fabricantes que deciden no incluir esta funcionalidad.
- Google puede decidir no distribuir una aplicación sino cumple con sus criterios, y puede censurar la aplicación dependiendo de su contenido.
- Debes pagar una cantidad de 25$ dólares para publicar tu aplicación, y Google se quedara con el 30% de tus beneficios.

Algunas de las alternativas disponibles para publicar tus aplicaciones son las siguientes:

- **Amazon AppStore:** Es la plataforma de distribución de aplicaciones de Amazon, publicar una aplicación esta tienda nos permitirá que los usuarios de la plataforma Kindle de Amazon puedan utilizar nuestras aplicaciones, ya que estos dispositivos no dan soporte a la Play Store. Se requiere un pago de 99$ al año para publicar aplicación después del primer año de uso, y permite aplicaciones de pago.

- **Samsung Apps:** Creada por la empresa Samsung, está disponible en todos los dispositivos Samsung por defecto. Como la amazon Appstore no requiere un pago para publicar aplicaciones, y permite aplicaciones de pago.
- **SlideMe:** Esta tienda de aplicaciones como cualidad destacada, es que tiene un sistema muy potente para que los usuarios puedan descubrir aplicaciones dependiendo de su situación geográfica, o las aplicaciones utilizadas por el usuario. Tampoco requiere ningún pago inicial.
- **Aptoide:** Es una plataforma de distribución gratuita, y de código libre, sin publicidad, y sin ningún tipo de rastreo de información de los usuarios. Cada desarrollador maneja su propia tienda, no es un sistema centralizado. Está basado en APT (Linux package manager). No requiere ningún pago para publicar aplicaciones.
- **F-droid**: Está basado en el código de Aptoide, y solo se diferencian en el formato usado en los paquetes de las aplicaciones, y otros cambios estéticos. No requiere ningún pago para publicar aplicaciones, solo permite aplicaciones gratuitas y de código libre.
- **Getjar:** Es uno de los mercados de aplicaciones más antiguo, y es multiplataforma, no siendo exclusivo para Android. No permite aplicaciones de pago.

NOMBRE	APLICACIONES DISPONIBLES	CANTIDAD DE DINERO PARA EL DESARROLLADOR	PAGO PARA PUBLICAR APLICACIONES
Play Store	1.000.000	70%	25$ pago único
Amazon App Store	100.000	70%	99$/año, primer año gratis
Samsung Apps	13.000	70%	Gratis
SlideMe	14.000	80-98% (depende del método de pago)	Gratis
Aptoide	100.000	Hasta el 87.5%	Gratis
F-droid	1.000	Solo aplicaciones gratuitas	Gratis
Getjar	257.000	Solo aplicaciones gratuitas	Gratis